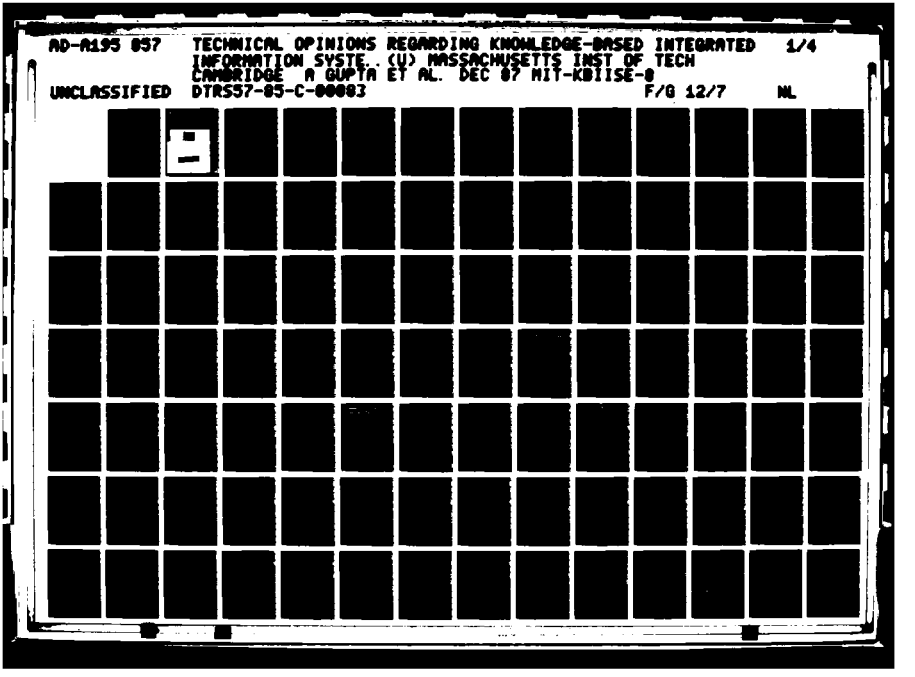
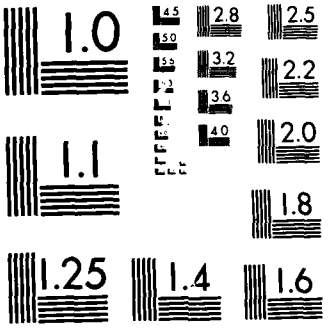

AD-A195 857 TECHNICAL OPINIONS REGARDING KNOWLEDGE-BASED INTEGRATED 1/4
INFORMATION SYSTE. (U) MASSACHUSETTS INST OF TECH
CAMBRIDGE A GUPTA ET AL. DEC 87 MIT-KBIISE-8
UNCLASSIFIED DTRS57-85-C-00083 F/G 12/7 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Massachusetts
Institute of
Technology

Knowledge-Based
Integrated Information
Systems Engineering
(KBIISE) Project

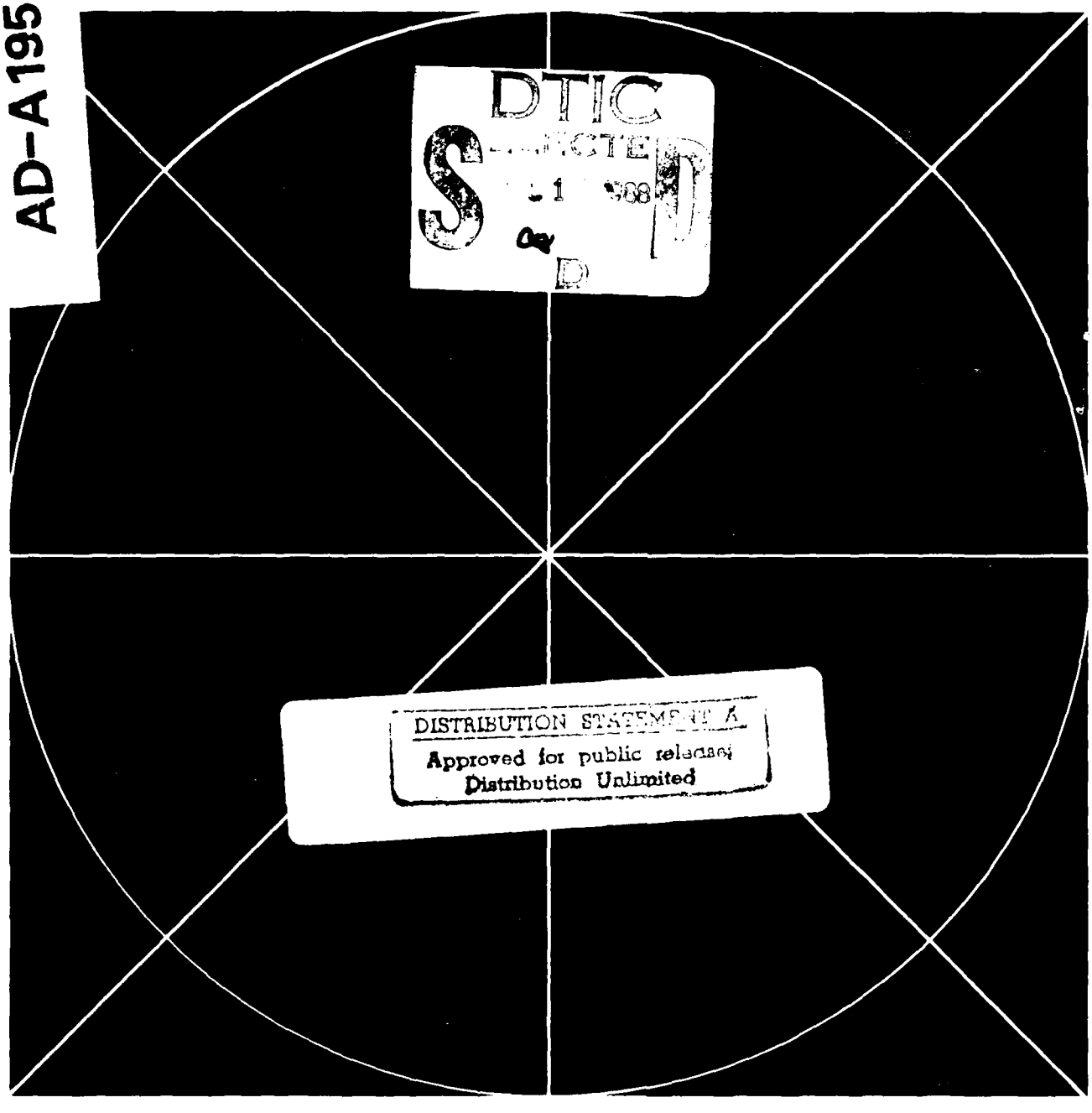
Volume 8

Technical Opinions Regarding Knowledge-Based Integrated Information Systems Engineering

(2)

Amar Gupta
Stuart Madnick
SERIES EDITORS

AD-A195 857



DTIC
S ELECTE D
JUL 1 1988
CU
D

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER KBIISE-8	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Technical Opinions Regarding Knowledge-Based Integrated Information Systems Engineering		5. TYPE OF REPORT & PERIOD COVERED Part of final report; Sept 86- Jan 88 6. PERFORMING ORG. REPORT NUMBER KBIISE-8
7. AUTHOR(s) Amar Gupta and Stuart Madnick (editors)		8. CONTRACT OR GRANT NUMBER(s) DTRS57-85-C-00083
9. PERFORMING ORGANIZATION NAME AND ADDRESS Massachusetts Institute of Technology Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Transportation Systems Center Broadway Street Cambridge, MA 02142 (In conjunction with USAF)		12. REPORT DATE December, 1987 13. NUMBER OF PAGES 350 pages
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Transportation Systems Center Broadway Street Cambridge, MA 02142		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE na

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release: distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This volume is one of eight volumes prepared by MIT for the Department of Transportation and Department of Defense (US Air Force).

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Integration, knowledge, databases, systems engineering, methodologies, information modeling, heterogeneous database systems.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This volume encapsulates the views of a panel of experts drawn from government, industry, and academia. It is divided into three parts. The first part, "Record of Discussion Held at the First Meeting of Technical Advisory Panel on February 17, 1987," highlights the most significant dimensions of Knowledge-Based Integrated Information Systems Engineering.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

The second part, "Record of Discussions held at the Second Meeting of Technical Advisory Panel on May 21 and May 22, 1987," presents opinions in the areas of: (a) Organization, Strategy, and Management; (b) Distributed Database Technology; (c) Knowledge and Semantics; and (d) Information Modeling and Mapping. A framework for tackling the problems in these areas is also presented.

The third part, "Contributions by Members of the Technical Advisory Panel (TAP)" contains ten outstanding papers contributed by members of the TAP. The first set of three papers focuses on distributed databases. The second set of three papers concentrates on information modeling alternatives. The final set of four papers examines a range of auxiliary issues in the context of large-scale, distributed heterogeneous information systems.

TECHNICAL OPINIONS REGARDING KNOWLEDGE-BASED INTEGRATED INFORMATION SYSTEMS ENGINEERING

Amar Gupta
Stuart Madnick

Series Editors

Knowledge-Based Integrated Information Systems Engineering (KBIISE) Report: Volume 8

Massachusetts Institute of Technology

DTIC
COPY
INSPECTED
6

Accesion For
NTIS CRA&I
DTIC TAB
Unannounced
Justification
By
Distribution /
Availability Codes
Dist Avail and / or Special
A-1

TECHNICAL OPINIONS REGARDING KNOWLEDGE-BASED INTEGRATED INFORMATION SYSTEMS ENGINEERING

About This Volume

This volume encapsulates the views of a panel of experts drawn from government, industry and academia. It is divided into three parts. The first part, "Record of Discussions Held at the First Meeting of Technical Advisory Panel on February 17, 1987," highlights the most significant dimensions of Knowledge-Based Integrated Information Systems Engineering.

The second part, "Record of Discussions held at the Second Meeting of Technical Advisory Panel on May 21 and 22, 1987," presents opinions in the areas of:

(a) Organization, Strategy, and Management;

(b) Distributed Database Technology;

(c) Knowledge and Semantics; and

(d) Information Modeling and Mapping.

A framework for tackling the problems in these areas is also presented.

The third part, "Contributions by Members of the Technical Advisory Panel (TAP)", contains ten outstanding papers contributed by members of the TAP. The first set of three papers focuses on distributed databases. The second set of three papers concentrates on information modeling alternatives. The final set of four papers examines a range of auxiliary issues in the context of large-scale, distributed, heterogeneous information systems.

Table of Contents

	Page
SERIES EDITORS' NOTE	1
RECORD OF DISCUSSIONS HELD AT THE FIRST MEETING OF THE TECHNICAL ADVISORY PANEL ON FEBRUARY 17, 1987 (Technical Report #2)	5
RECORD OF DISCUSSIONS HELD AT THE SECOND MEETING OF THE TECHNICAL ADVISORY PANEL ON MAY 21 AND 22, 1987 (Technical Report #23)	21
CONTRIBUTIONS BY MEMBERS OF THE TECHNICAL ADVISORY PANEL (TAP) (Technical Report #24)	53

Knowledge-Based Integrated Information System Engineering Project: Volume 8
Amar Gupta and Stuart E. Madnick, Editors

Copyright © Massachusetts Institute of Technology, 1987.

DEDICATED

TO

THE

NEXT

GENERATION

OF

PROFESSIONALS

SERIES EDITORS' NOTE

This book is one of eight volumes published by MIT as part of the Knowledge-Based Integrated Information Systems Engineering Project (KBIISE). In order to appreciate the papers in this book, it is necessary to be aware about the theme of the KBIISE project, its major objectives, and the different documents that summarize the research accomplishments to date.

Goal

The primary goal of the KBIISE project is to integrate islands of disparate information systems that characterize virtually all large organizations. The number and the size of these islands has grown over years and decades as organizations have invested in an increasing number of computer systems to support their growing reliance on computerized data. This has made the problem of integration more pronounced, complex, and challenging.

The need for multiple systems in large organizations is dictated by a combination of technical reasons (such as the desired level of processing power and the amount of storage space), organizational reasons (such as each department obtaining its own computer based on its function), and strategic reasons (such as the level of reliability, connectivity, and backup capabilities). Further, underlying trends in the information technology area have led to a situation where most organizations now depend on a portfolio of information processing machines, ranging from mainframes to minicomputers and from general purpose workstations to sophisticated CAD/CAM systems, to support their computational requirements. The tremendous diversity and the large size of the different systems make it difficult to integrate these systems.

Key Participants

The above problem is becoming increasingly evident in all large government agencies and in large development programs. In the fall of 1986, the U.S. Air Force (USAF) and the Transportation Systems Center (TSC) of the U.S. Department of Transportation approached M.I.T. to conduct and to coordinate research activity in this area in order "to develop the framework for a comprehensive methodology for large scale distributed, heterogeneous information systems which will provide: (i) the necessary structure and standards for an evolving top down global framework; (ii) simultaneous bottom up systems development; and (iii) migratory paths for existing systems."

Both USAF and TSC provided sustained assistance to members of our research team. In addition, Citibank and IBM provided some funds for research in very specific areas. One advantage of our corporate links was the opportunity to analyze and to generate case studies of actual decentralized organizational environments.

The research sponsors and MIT agreed that in order to deal with the heterogenity issue in a meaningful way, it was important that a critical mass of influential individuals participate in the development of solutions. Only through widespread discussion and acceptance of a proposed strategy would it become feasible to deal with the major problems. For these reasons, a Technical Advisory Panel (TAP) was constituted. Nominees to the TAP included experts from academic and research organizations, government agencies, computer companies, and other corporations. In addition, several subcontractors, the primary one being Texas A&M University, provided assistance in specific areas.

Technical Outputs

The scope of the work included (i) technical issues; (ii) organizational issues; and (iii) strategic issues. On the basis of exploratory research efforts in all these areas, 24 technical reports were prepared. Eighteen of these reports were generated by MIT research personnel, and their respective areas of investigation are summarized in the figure on the opposite page.

The five technical reports, not represented in the figure, are as follows:

#1. Summary.

#2. Record of discussions held at the first meeting of the Technical Advisory Panel (TAP) on February 17, 1987.

#3. Consolidated report submitted by Texas A&M University.

#21. Annotated Bibliography.

#23. Record of discussions held at the second meeting of the Technical Advisory Panel (TAP) on May 21 and 22, 1987.

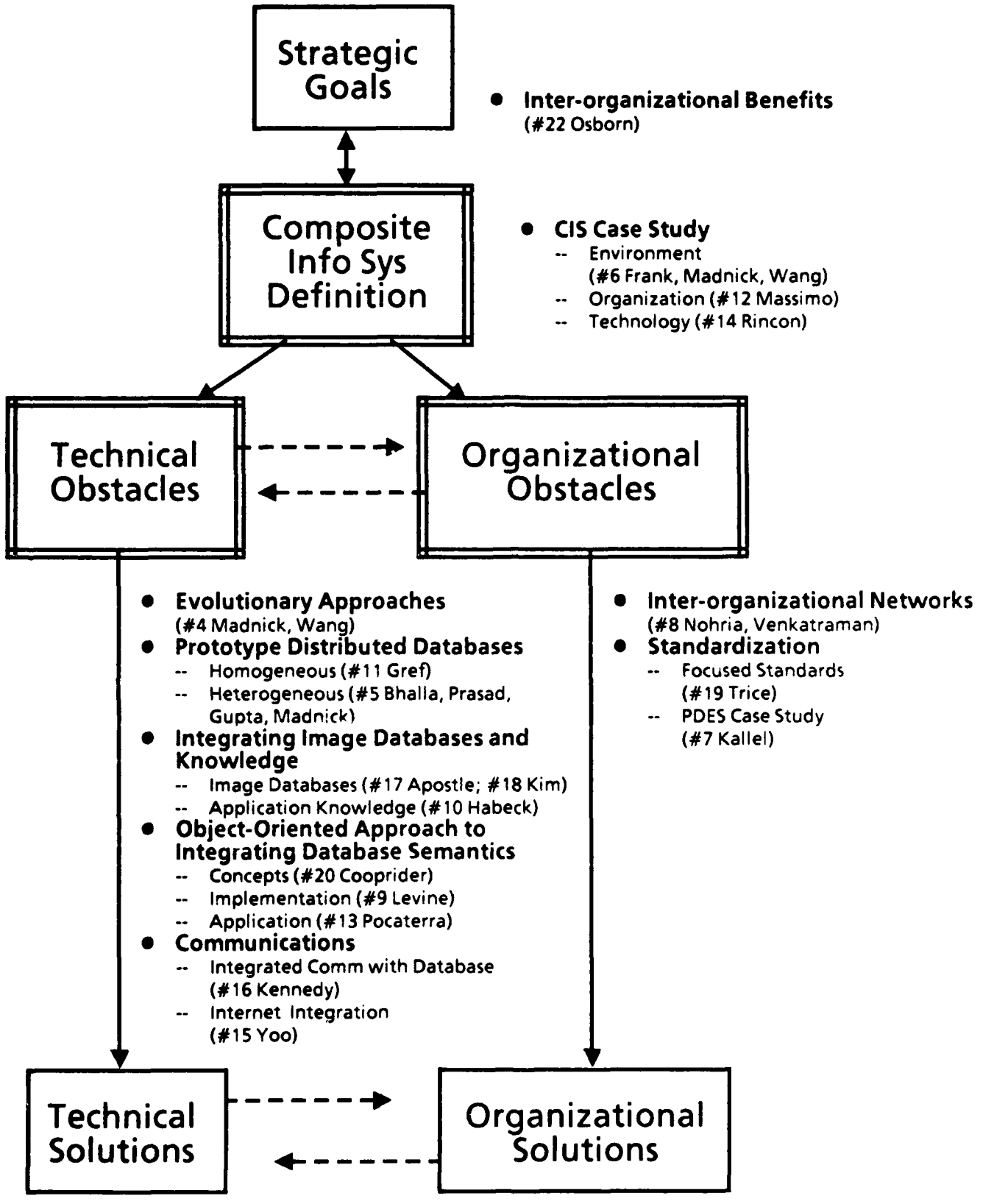
#24 Contributions received from members of the TAP highlighting their views on various aspects of the problem.

All the 24 technical reports have been edited and reorganized as an eight-volume set. The titles of the different volumes are as under:

1. KNOWLEDGE-BASED INTEGRATED INFORMATION SYSTEMS ENGINEERING– HIGHLIGHTS AND BIBLIOGRAPHY
2. KNOWLEDGE-BASED INTEGRATED INFORMATION SYSTEMS DEVELOPMENT METHODOLOGIES PLAN
3. INTEGRATING DISTRIBUTED HOMOGENEOUS AND HETEROGENEOUS DATABASES – PROTOTYPES
4. OBJECT-ORIENTED APPROACH TO INTEGRATING DATABASE SEMANTICS
5. INTEGRATING IMAGES, APPLICATIONS, AND COMMUNICATIONS NETWORKS
6. STRATEGIC, ORGANIZATIONAL, AND STANDARDIZATION ASPECTS OF INTEGRATED INFORMATION SYSTEMS
7. INTEGRATING INFORMATION SYSTEMS IN A MAJOR DECENTRALIZED INTERNATIONAL ORGANIZATION
8. TECHNICAL OPINIONS REGARDING KNOWLEDGE-BASED INTEGRATED INFORMATION SYSTEMS ENGINEERING

Volume 2 contains the report submitted by Texas A&M and Volume 8 highlights the views of members of the TAP. Activities described in the other 6 volumes have been conducted at MIT.

EXPLORATORY RESEARCH EFFORTS



Acknowledgments

Funds for this project have been provided by U.S. Air Force, U.S. Department of Transportation (Contract Number DTRS57-85-C-00083), IBM, and Citibank. We thank all these organizations and their representatives for their support. In particular, we are indebted to Major Paul Condit of U.S. Air Force for his initiative in sponsoring this project, to Dr. Frank Hassler, Bud Giangrande, and Bob Berk of the Transportation Systems Center (TSC) for their support and assistance, to Professor Joseph Sussman, Director, Center for Transportation Studies (CTS) at MIT for his help and encouragement, and to all the individuals whose results have been published in this book.

We would welcome receiving feedback from readers of this book.

Amar Gupta and S.E. Madnick
Massachusetts Institute of Technology
Cambridge, Massachusetts.

RECORD OF DISCUSSIONS HELD AT THE FIRST MEETING OF TECHNICAL ADVISORY PANEL ON FEBRUARY 17, 1987

AMAR GUPTA AND S.E. MADNICK

Under the aegis of a research contract awarded by U.S. Air Force and Transportation Systems Center, the Knowledge-Based Integrated Information Systems Engineering Project was initiated at MIT in fall 1986. Subsequently, it was agreed that this project should seek the advice of a panel of experts drawn from government, industry, and academia.

At the first meeting of the Technical Advisory Panel (TAP), the areas considered most significant were as follows:

(a) Knowledge/Semantics
  (i) Development and Standardization of Generic Semantic Model
  (ii) Extensions of E-R Model for Heterogeneous Environments
  (iii) Better Tools for Domain Modeling and Knowledge Representation
  (iv) Mapping of Process into Information Model
  (v) Semantics for Advanced DBMS

(b) Evolution
  (i) Update Legacy of Systems
  (ii) Life Cycle of Systems
  (iii) Development of Framework before Details

(c) Performance Feasibility
  (i) Operational Feasibility and Simulation
  (ii) Performance for Update Concurrency

(d) Organizational and Management
  (i) Local versus Global Control
  (ii) Management Awareness and Extensability
  (iii) Education of Management to Understand Process
  (iv) Multi-level "Federated" Systems

The motivation for suggesting the above areas, and the opinions of each TAP member, are reflected in this report.

TECHNICAL REPORT #2

I. PARTICIPANTS

The first meeting of the Technical Advisory Panel (TAP) was held at M.I.T. on February 17, 1987. The individuals who participated were as follows:

(A) Members of Technical Advisory Panel

1.	Daniel S. Appleton	DACOM
2.	Howard Bloom	National Bureau of Standards
3.	Robert R. Brown	Rand Corporation/Consultant
4.	Peter P. Chen	MIT/Louisiana State University
5.	Lynette Hirschman	Unisys
6.	Ellen Knapp	Coopers and Lybrand
7.	Sam Nusinow	Knowledge Base Engineering Ent.
8.	David Reiner	Computer Corporation of America
9.	Charles Standridge	A. Pritsker and Associates
10.	Paul Thompson	Control Data Corporation
11.	Gio Wiederhold	Stanford University

(Bill Putnam of Rockwell International, John Machado of SPAWAR, and Ray Liuzzi of RADC could not attend this meeting. Complete addresses, along with telephone numbers of all TAP members, are enclosed as Appendix A.)

(B) Representatives of the U.S. Air Force

12. Major Paul Condit
13. Mark Hoffman

(C) Representatives of Transportation Systems Center

14. Frank Hassler
15. Bud Giangrande
16. Bob Berk

(D) Representatives of M.I.T.

17. Abraham J. Siegel
18. Joseph Sussman
19. Tom Magnanti
20. S. E. Madnick
21. Amar Gupta
22. Y. Rich Wang
23. B.E. Prasad
24. Ali Tobah
25. Ajit Kimbal
26. Manuel Quintana
27. Maher Kallel

28. Ted Hennessy
29. Lloyd Brodsky
30. Marina Pocaterra

(E) Representatives of Texas A&M University

31. Ric Mayer
32. Pat Friel

The meeting was organized by Massachusetts Institute of Technology, in consultation with representatives of the research sponsors (U.S. Air Force and Transportation Systems Center).

II. OBJECTIVES

In virtually all governmental agencies, multinational corporations, and other large organizations, there is a growing realization of the urgent need to integrate large volumes of information stored on multiple, independent systems. While there is widespread consensus about the problem, very little effort has so far been devoted towards finding solutions to the problem. In order to bridge this gap, MIT, Air Force, and TSC jointly agreed to constitute a Technical Advisory Panel (TAP) of experts drawn from government, industry, and academia.

For its first meeting, the objectives of the TAP were defined to be as follows:

(1) To describe the problem area in the context of the Air Force and other large organizations;
(2) To gain insights into the problems faced by individuals responsible for developing and implementing methodologies for large scale information management;
(3) To learn about current endeavors, both by researchers as well as by individuals in government and industry;
(4) To become familiar with new standards being established and contemplated, for facilitating exchange of information and knowledge;
(5) To discuss potential solutions to the problem;
(6) To define mechanisms and channels by which TAP members could provide ideas to the research team on a continuing basis; and
(7) To discuss any other issues of interest to TAP members.

A copy of the agenda is placed at Appendix B. Each TAP member presented his or her ideas in approximately 20 minutes. After each presentation, there was a question-answer session of about 10 minutes per speaker.

III. PREPARATORY COMMENTS

The meeting began with opening remarks by the following:

(1) Professor Abraham J. Siegel, Dean, Sloan School of Management, M.I.T.;
(2) Dr. Frank Hassler, Assistant Director, Office of Information Resources, Transportation Systems Center;
(3) Professor Joseph Sussman, Director, Center for Transportation Studies, M.I.T.; and
(4) Professor Thomas L. Magnanti, Area Head, Management Science, Sloan School of Management, M.I.T.

It was mentioned that Sloan School of Management is currently involved in several large research efforts directed towards better utilization of information technology. The objectives of the Center for Information Systems Research (CISR) and Management in the Nineties Project were described in this context.

Immediately following the opening remarks, Major Paul Condit described the CALS (Computer Aided Logistics Support) initiative, and its major objectives of:

(1) Accelerating integration of R&M design tools into contractor CAD/CAE systems;
(2) Automating contractor processes for generating technical information; and
(3) Increasing rapidly DOD capability for receiving, distributing, and using technical information in digital form.

Unlike the present position of using dissimilar and incompatible media, such as paper, tape, and microfiche, the endeavor is towards developing an electronic common product database that could be efficiently shared by engineering, operations, and logistics.

Major Condit next described the major highlights of the Integrated Design Support (IDS) System, which specifically focuses on the issues of capture, management, and communication of technical data from the initial design stage through logistic

operations, in fact, over the entire life cycle of the weapon system. He emphasized the following facts:

(1) The Air Force and its contractors generate and utilize many types of technical information such as:
   (a) Engineering data (drawings, specifications, product definition data, etc);
   (b) Technical documentation data (technical orders, manuals, materials, software documentation, etc.);
   (c) Logistics Planning Data (R & M requirements, maintenance task analysis, personnel, facilities, etc.).

(2) There is a growing trend toward using microcomputers and minicomputers, expanding into new application areas (CAD, CAM, etc.), and using data in real time and near real time basis; and

(3) Technical Information System Development has generally taken place in an uncoordinated and uncontrolled manner.

In view of the above facts, it has become very critical to develop a comprehensive and disciplined methodology and structure for integrated information systems engineering.

Major Condit concluded his presentation by describing the requirements as follows:

(1) Developing a comprehensive methodology for large scale distributed, heterogeneous information systems development;
(2) Identifying structure and standards for:
   (a) Evolving top down global framework (planning and logical design);
   (b) Simultaneous bottom up systems development and implementation;
   (c) Migration paths for existing systems;
(3) Developing an automated and integrated suite of methods, techniques, and tools to support the methodology.

Next, Professor S. E. Madnick described the Strategic Applications, Technology, and Organization Research Initiative (SATORI). In order to achieve success, it is necessary that three fundamental ingredients be present. These are: (i) strategy; (ii) organization; and (iii) technology. It is impossible to achieve success if any of three factors is missing

or lacking. Broad concerns in the three areas are as follows:

(1) Strategic Applications:
    (a) Product differentiation;
    (b) Cost effectiveness;
    (c) Value-added.

(2) Organizational:
    (a) Organizational structure
    (b) Role
    (c) Leverage individual
    (d) Group activities
    (e) Coordination

(3) Technology:
    (a) Database
    (b) Communications
    (c) Expert Systems
    (d) Connectivity
    (e) Interfaces

Professor Madnick also described the overlap between Knowledge, Information, Connectivity, and Interface aspects. Next, he discussed the concept of Past Systems, Future Imperfect Systems, and Future Perfect Systems, and the characeristics of systems in each of these categories.

Dr. Amar Gupta then described the key milestones relating to this project. He said that the project was initiated in the fall of 1986, and the first phase was expected to be completed in August 1987. In this phase, the emphasis is on identification of alternative approaches and formulation of recommendations. This phase consists of four steps as follows:

Step 1: Initiation of Project
Step 2: Identification of Alternative Approaches
Step 3: Discussion of Alternative Approaches
Step 4: Formulation of Recommendations.

The TAP meeting on February 17, 1987 falls under the set of activities defined for Step 2 above. A subsequent meeting would be arranged to discuss alternative approaches (Step 3). It was emphasized that, apart from ideas presented at formal meetings, input from TAP members would be welcome at all stages.

IV. PRESENTATIONS BY INVITEES-- FIRST SESSION
(10:15 a.m. - 12:15 p.m.)

(1) Howard Bloom, National Bureau of Standards

Mr. Bloom described the efforts being done in-house, as well as NBS sponsored activities being conducted at other places, in particular, at the University of Florida.

NBS has built a composite information system that encompasses all manufacturing functions: design, process planning, machine tool programming and loading, and inspection. They have had a prototype system operating since late 1986, which is available to universities and manufacturers.

The flexible manufacturing prototype system provides integration in a heterogeneous environment in which products from different vendors have been used. The system architecture is based on local control of functional databases which are linked through the IMDAS (Integrated Manufacturing Data Administration System). IMDAS is responsible for providing access to data through a project wide language as well as for translation of information between existing commercial products and the neutral access interface.

IMDAS is a distributed data system linking factory, engineering, and management. Multiple layers of data base control integrate progressively larger segments of the system, from operations within a "workstation" to overall system control.

Mr. Bloom mentioned that major research issues of interest to AMRF (Automated Manufacturing Research Facility) are:

(a) Distributed data management
(b) Generic Data Model
(c) Command translators
(d) Development of standards.

At present the three main areas of thrust are (i) Knowledge Based Engineering; (ii) Databases; and (iii) Geometric modeling.

Mr. Bloom also commented on the Rapid Acquisition and Manufacturing of Parts (RAMP) program sponsored by the navy. Currently, procurement is hampered not by manufacturing times, but by the procurement cycle. The new automatic bidding system will contain all bids on-line. Suppliers will have the flexibility of changing information on a periodic basis. The procurement time is likely to reduce from the current level of

200 days to 30-60 days. As manufacturing requirements change, there is a need for a new set of time-critical operations, which in turn imposes additional requirements on the AMRF approach.

(2) Gio Wiederhold, Stanford University

Professor Wiederhold began his presentation by differentiating between data and knowledge. The former is a set of facts, while the latter involves semantics; the former is dynamic in nature, while the latter changes at a much slower pace.

Existing data systems contain both knowledge and data in a variety of mixes. Normalization yields: (i) Non-redundant data; and (ii) Structured knowledge, constraints, and processing rules. For example, engineering information systems can be decomposed to segregate data from knowledge.

The procedure for integration of several data systems must first deal with the issue of recognizing differences in the scope of domains. For example, the definition of an employee may differ between the personnel department and the payroll department. Common standards cannot be enforced in federated systems, wherein each participating system has evolved independently of others.

In federated systems, it becomes imperative to deal with asynchrony. The personnel department may be updating its records on a daily basis, while the payroll department may be operating on a weekly or monthly schedule. This makes virtually all conventional concurrency control mechanisms irrelevant. Instead, it becomes necessary to think of updates in terms of primary copies and secondary transactions. The former are generated at the time of occurrence. However, delays in secondary transactions can be tolerated as long as they are received prior to the next operating cycle for that particular system.

Structured design of large databases would involve new design approaches. Current applications are procedure-based, and programs abound with control structures such as IF statements. These structures must be replaced by rule-based approaches. Advanced systems also require set-based accesses, and efficient dataflow mechanisms. In data engineered design, an approach based on the semantics of data may be better than a top-down design starting from a procedural objective.

Professor Wiederhold concluded with the remarks that a solution to the problem of large knowledge intensive information systems involves (i) Understanding; (ii) Decomposition; and (iii) Structure based approach focusing on high demand data and high complexity knowledge.

(3) Daniel Appleton -- DACOM

Mr. Appleton began his presentation with a list of past and current efforts in the area of distributed heterogeneous systems. This list included:

(a) ICAM CBIS
(b) ICAM IISS
(c) MULTIBASE
(d) CIM DATA ENGINE
(e) IMDAS
(f) IDS
(g) CALS

He mentioned that CALS is undoubtedly the largest project.

Mr. Appleton described the link between the sending user and the receiving user in terms of (i) Presentation; (ii) Data; (iii) Network Transaction Manager, and (iv) Communications. The term "data" includes semantics of the data.

Instead of striving towards a global solution for all possible scenarios, he emphasized the need for "scoping" the problem in terms of the following dimensions:

(a) Mode -- Exchange/On-line Data Access;
(b) Data Type -- Numeric/Alphanumeric/Voice/Image;
(c) Machines -- Homogeneous/Heterogeneous;
(d) DBMS -- None/Hierarchical/Network/Relational;
(e) Access -- Update/Retrieval;
(f) Environment -- Local Area Network/Wide Area Network;
(g) Security -- Light/Tight

As far as (e) is concerned, a situation involving updates poses a geometric increase in difficulty as compared to an equivalent "retrieval-only" situation.

Mr. Appleton felt that relevant data environment issues include:

(a) Structures (definitions and constraints);

(b) Types;
(c) Volumes;
(d) Consistency;
(e) Accessibility;
(f) State Management;
(g) Extensibility;
(h) Flexibility; and
(i) Levels of Abstraction

In addition, he briefly discussed the great difficulty involved in modeling existing systems realistically, and the need to calibrate expectations from the systems implemented.

(4) David Reiner - Computer Corporation of America

Dr. Reiner first described his interpretation of the field of Knowledge-Oriented Information Management. He felt that the more the DBMS "understands" about application semantics, the more scope it has for optimization of data access and data combination.

Current DBMS technology is inefficient in terms of its handling of many information types such as maps, drawings, photographs, text, time, signals, and use defined types. Apart from these areas, it is desirable to consider enhancing DBMS techniques to handle abstract data type and to offer superior dimensional semantics capabilities. Also, it would be appropriate to incorporate situation-action rules for expressing both domain knowledge and problem solving expertise. Newer versions of DBMS should provide for:

(a) Flexible view update translation (propagation rules);
(b) Flexible integrity control (how to proceed after a violation is detected);
(c) Flexible concurrency control (including relaxation of serializability);
(d) Version control (especially for CAD/CAM applications); and
(e) Network partition recovery (consistency maintenance).

The driving goal is to support the illusion that an integrated database exists, which combines information from multiple sources, grouped as desired by the user. In order to achieve this integration, it is necessary to develop a single global data manager, format, and data model. It is also necessary to define a virtual database that matches corresponding objects from different sources and resolves inconsistencies.

This in turn involves defining formal operators for combining and grouping information in desired ways, and developing new and better query processing strategies.

Dr. Reiner described current activities at CCA in the above areas. He described Multibase in terms of its capabilities for:

(a) Schema integration;
(b) Data incompatibility handling;
(c) Query optimization; and
(d) Query translation.

He mentioned various extensions to Multibase, in the areas of conflict resolution rules, natural langrage front-end, and view integration expert sub-systems. He concluded with the observation that performance is a major issue, and it is usually not appreciated that information retrieved in 4-5 minutes using Multibase would have required maybe a year's time to be retrieved by other means.

V. PRESENTATIONS BY INVITEES - SECOND SESSION (1 p.m. - 3 p.m.)

(1) Peter Chen -- MIT/Louisiana State University

Professor Chen focused his presentation on how ERA (Entity-Relationship Approach) can be expanded to deal with the situation of large, heterogeneous computing environments.

He emphasized that the ER methodology provided the foundations for representing data as well as knowledge. Extensions to ERA to handle knowledge have included incorporation of the following:

(a) "IS-A" links
(b) Set-operation relationship
(c) Decomposition relationship
(d) Existence-dependence and identifier-dependence concepts
(e) Precedence relationship
(f) Total and Partial relationship
(g) Conversion of relationships into high-level entities
(h) Time dimension
(i) Logic ER model
(j) New structures

He described various attempts in the U.S. as well as abroad to use and extend ER models for new application environments.

Prof. Chen felt that ER models can serve as the basis for:

(a) Managing heterogeneous computing environments
(b) Integrating models
(c) Integrating methodologies
(d) Integrating tools; and
(e) Integrating data and knowledge.

(2) Ellen Knapp -- Coopers and Lybrand

Ms. Knapp first described the enormity of the CALS project. It is a multi-service endeavor. The Navy and Air Force each have over 30 projects under the CALS umbrella. Among the Navy projects, there is one project that alone involves a procurement of $1.2 billion of hardware for CAD equipment. Given its enormous set and the large set of players involved, it is imperative to plan carefully.

She emphasized that plans must be based on business requirements. The business model must serve as the foundation for building the data architecture, which in turn should be used for defining the information systems architecture. This architecture should then be delineated in terms of hardware, software, and communications. The most important component of the technology plan is the data architecture. If one attempts to define the component architectures before clearly finalizing the business model, then one is in essence having "a solution in search of a problem!"

Ms. Knapp also made the following suggestions:

(a) In order to solve the problem it is necessary to define its bounds;
(b) Whatever methodology is selected, be sure that:
    (i) it can be applied to business; and
    (ii) it can be extended (real world is not just five guys!)
(c) Methodologies are more or less stable but tools and procedures are always changing;
(d) Look at information system engineering with an international perspective (for U.S. business to survive); and
(e) For military application, if access is allowed, then who owns the database?

(3) Robert Brown - Consultant

Mr. Brown commenced his presentation with "the meaning of symbols" and their relevance in human knowledge and communications. He distinguished between real world, concept world, and symbol world, and emphasized the fact that, given these mappings, inconsistencies between databases should be regarded as a natural phenomenon. In fact, "any database containing more than 10 records invariably contains some inconsistencies."

Mr. Brown distinguished between various stages of information modeling:

(a) Early data models which contained no semantic base;
(b) E-R models which were a step in the right direction;
(c) More recent models such as ELKA which are fairly advanced but still limited.

He felt that the goal of a semantic model should be to establish a framework for analyzing and describing the meanings of symbols used by humans for communication. Ideally, it should be feasible to understand the semantics simply by browsing through the database.

Information modeling is a process for producing a semantic model. Unfortunately, there is no agreement on either the process or the model. For example, in the case of textual documents, there are two approaches: SGML (Standard Generic Markup Language) versus WYSIWYG (what you see is what you get). The former has been standardized, but is still flawed and virtually unreadable; the latter is appealing to look at, but useless for purposes of database.

Mr. Brown concluded with the observation that the key issue is to develop and standardize on a generic semantic model, along with associated tools (software) to support its use. Also, it should be realized that documents are currently used for control purposes; if the documents are removed, what will happen to the control procedures?

(4) Sam Nusinow - Knowledge Based Engineering Enterprises

Mr. Nusinow observed that the current problem has been caused by several factors:

(a) Media Inconsistency -
Technical data resides on paper, microfilm, and electronic media. In order to automate, product data is currently at a transition stage, requiring it to be duplicated across different media types:

(b) Application Incompatibility -
Process and product data configuration management and control procedures vary even between units within a single company. In particular, each CAD/CAM application may use its own particular format. Global knowledge about product data and product data cycles is not shared. As such, applications do not effectively manage and control the product data evolution;

(c) Automated Configuration Management and Control (CM&C) Inadequacy -
Few systems provide all of the functions that are required to conduct automated CM & C effectively. Further, none of the automated CM & C systems effectively address the distributed heterogeneous database issue;

(d) Process/Product Data Analysis Complexity -
There is a lack of integrated system engineering tools that can effectively support process/product data modeling and collection, and transformation for subsequent storage in a knowledge base; and

(e) Semantic Data Engineering Tool Insufficiency -
No tools are currently available that can effectively collect, define, analyze, and maintain the semantics, rules, and CM & C procedures associated with product data.

He concluded with the observation that it will continue to be a time-consuming and costly task to implement CM & C capabilities within the many different engineering and manufacturing environments, unless a comprehensive move is made to mitigate the problems described above.

VI. PRESENTATIONS BY INVITEES - THIRD SESSION
(3 p.m.--5 p.m.)

(1) Charles Standridge - Pritsker & Associates, Inc.

Mr. Standridge described the importance of simulation exercises. Further, he discussed the key points related to "SLAM II" Simulation System and "The Extended Simulation System "TESS"). The latter comprises of four interrelated modules for model development, analysis, presentation, and management of data respectively.

He spoke about recent advances in the area of simulation, and the incorporation of ideals from the realm of artificial intelligence. Goal directed simulation, for example, involves and execution monitor and an inference engine. Computers can assist in the presentation of results as well as in model derivation.

(2) Paul Thompson - Control Data Corporation

Mr. Thompson covered the twin issues of methodologies for integrated knowledge base/database applications and methodologies for large scale information management.

He distinguished between various levels of data models as follows:

(a) Linquistic/Logical Predicates;
(b) Binary Object role model;
(c) Conceptual data model; and
(d) Lower level models involving conceptual schema, internal schema, and external schema.

In order to suit needs at various levels, different types of models are appropriate. He mentioned three issues: (i) Ambiguity; (ii) Assumptions; and (iii) Different views that must be analyzed.

For example, at the top level of strategic information engineering, it is necessary to think about an entity model and a subject database plan. But at the level of tactical information engineering, it is more useful to consider a semantic information model and a neutral data model. Finally, at the level of system and data architecture engineering, one needs to consider a three-schema/view model and a physical database design.

Mr. Thompson briefly described the object role model. He emphasized the need to first agree on a framework or reference model.

(3) Lynette Hirschman - UNISYS

Dr. Hirschman discussed the subject of natural language database interfaces.

A natural language system should be able to analyze incoming messages and distinguish remarks from pro-forma message fields.

The former are used by the text processing component of the system, while the latter are used to update the database. In designing such a system, the following issues must be considered:

(a) Scanning versus in-depth understanding;
(b) Portability;
(c) Robustness;
(d) Habitability; and
(e) Knowledge acquisition.

Many of the above aspects are active areas of research.

In order to develop easy and efficient user interfaces for a large database environment, the following aspects require attention:

(a) Ability to address across multiple DBMS by separating back-end query formation and database access from natural language front end;
(b) Development of separate knowledge representation framework;
(c) Provision of knowledge acquisition tools;
(d) Support of larger grammar;
(e) Development and support of better tools for domain modeling and knowledge representation;
(f) Support of different environments, for example, spreadsheet, report generation, and voice;
(g) Development and integration of "deep" understanding strategies with techniques for scanning large amounts of information; and
(h) Creation of more robust systems with superior diagnostics capabilities.

(4) Ric Mayer - Texas A&M

Mr. Mayer described the evolution of methodologies in the context of the Air Force. He mentioned that Project 1701 completed methodology work in 1982, and that the IDEF methods have remained frozen since 1979. He felt that IDS and IISS have proven the usefulness of three schema approaches to the information integration problem in a heterogeneous system environment.

In his opinion, the present objectives were as follows:

(a) To identify the current state of the practice in integrated information system development methodologies;
(b) To analyze current automated support tools and

environments for planning, analysis, design, construction, and maintenance;
(c) To identify industry needs for additional methodologies and tools;
(d) To explore the application of knowledge based systems technology to the system development process;
(e) To examine existing methods for their formal semantics;
(f) To define the requirements for a neutral information representation language; and
(g) To establish the development plan and to prepare the advocacy material.

Mr. Mayer also reiterated the fact that there is significant demand from the industry for improved methods and tools, and also for standardization.

VII. RECOMMENDATIONS AND CONCLUSION

Each TAP member was requested to state a few issues which the member thought were of paramount importance in his or her veiw. The responses were as follows:

(a) Bloom - Performance, concurrency, and rules for updating information;
(b) Wiederhold- Structure knowledge;
(c) Appleton - Legacy and updates;
(d) Chen - Extending ER model to heterogeneous distributed environments;
(e) Knapp - Extensability; management of both awareness and concepts;
(f) Brown - Generic semantic model and support model;
(g) Nusinow - Lifecycle on data and constraints;
(h) Standridge- Opertionally feasible;
(i) Thompson - Development of framework before details; education of both management and the personnel responsible for impelmentation; tools to bridge models;
(j) Hirschman-Domain models.

Based on a closer examination of the above list, it was determined that all the issues can be broadly classified into four main categories:

(a) Knowledge/Semantics;
  (i) Generic Semantics (Bob Brown)
  (ii) Extensions for Heterogeneous (Peter Chen)
  (iii) Domain Models (Lynette Hirschman)
  (iv) Process <---> Information Model (Sam Nusinow)
  (v) DBMS Semantics (David Reiner)

(b) Evolution
  (i) Update Legacy of Systems (Dan Appleton)
  (ii) Life Cycle of Systems (Sam Nusinow)
  (iii)Framework before Details (Paul Thompson)

(c) Performance Feasibility
  (i) Operational Feasibility (Charles Standridge)
  (ii) Performance for Update Concurrency (Howard Bloom)

(d) Organizational and Management
  (i) Local versus Global Control (Howard Bloom)
  (ii) Management Awareness and Extensability (Ellen Knapp)
  (iii) Educate Management to Understand Process (Paul Thompson)
  (iv) Multi-level "Federated" Systems (Gio Wiederhold)

The above set of four important parameters, and associated sub-parameters, constituted the major suggestion of the TAP to the research team. TAP members were encouraged to provide additional insights on a continuing basis.

Based on mutual convenience, it was agreed that the next meeting of TAP be scheduled for May 21, 1987. (This meeting has been subsequently enlarged into a two-day event (May 21-22), in view of requests from several TAP members and other participants). At this meeting, various members of the research team will make presentations of their approaches to various parts of the problem. Also, new TAP members will present their views on this project.

APPENDIX A: ADDRESS LIST

Daniel S. Appleton, President (213) 546-7575
D. Appleton Co., DACOM
1334 Park View Avenue, Suite 220
Manhattan Beach, CA 90266

Mr. Howard Bloom (301) 975-3509
Chief, Factory Automation Systems Div.
National Bureau of Standards
Building 220, Room A-127
Gaithersburg, MD 20899

Dr. Robert Brown (714) 773-4050
2229 Via Caliente
Fullerton, CA 92633

Mr. Peter P. Chen (504) 388-2482
Department of Computer Science
Louisiana State University
Baton Rouge, LA 70808

Dr. Lynnette Hirschman (215) 648-7554
Technical Director, Logic Systems
UNISYS - Department of Research and Development
P.O. Box 517
Paoli, PA 19301

Ms. Ellen Knapp (202) 822-4000
Coopers and Lybrand
1800 M Street N.W.
Washington, D.C. 20036

Mr. Sam Nusinow (513) 434-1716
Knowledge Based Engineering Enterprises
240 Elmwood Drive
Centerville, OH 45459

Mr. David Reiner (617) 492-8860
Computer Corporation of America
4 Cambridge Center
Cambridge, MA 02142

Dr. Charles R. Standridge (806) 794-7696
5412 88th Street
Lubbock, TX 79424

Mr. Paul Thompson (612) 853-6060
Control Data Corporation
P.O. Box Zero
Minneapolis, MN 55440

Mr. Gio Wiederhold (415) 723-0685
Stanford University
Margaret Jacks Hall, Room 436
Palo Alto, CA 94305

APPENDIX B

KNOWLEDGE-BASED INTEGRATED INFORMATION SYSTEMS ENGINEERING PROJECT

AGENDA FOR
TAP MEETING ON FEBRUARY 17, 1987
TO BE HELD IN THE PENTHOUSE OF THE MIT FACULTY CLUB, 50 MEMORIAL DRIVE, CAMBRIDGE, MA 02139

8:45 to 9:15 **Faculty Club Lounge**
Informal discussions, coffee and muffins

9:15 to 9:55 **Opening Remarks**
Dean Abraham J. Siegel, MIT
Frank Hassler, TSC
Joseph Sussman, CTS-MIT
Tom Magnanti, MIT

9:55 to 10:30 **Project Objectives**
AF - Major Paul Condit
MIT - S.E. Madnick/Amar Gupta

10:30 to 12:20
(i) Howard Bloom - Research efforts at and sponsored by NBS
(ii) Gio Wiederhold - Coordination problems in distributed environments
(iii) Daniel Appleton - Data integration in heterogeneous computing environments
(iv) David Reiner - Distributed databases and tools for designing databases
(v) Peter Chen - ER models in heterogeneous computing environments

12:20 to 1:20 **Lunch at Faculty Club** (Private Dining Room)

1:20 to 2:40
(i) Ellen Knapp - Analyzing and developing techniques for efficient information management
(ii) Bob Brown - Information modeling - theory and practice
(iii) Sam Nusinow - Integrated information engineering
(iv) Charles Standridge - Simulation

2:40 to 3:00 **Coffee Break**

3:00 to 3:40
(i) Paul Thompson - Methodologies for large scale information management
(ii) Lynette Hirschman - Natural language interfaces

3:40 to 5:00 **Discussion.**

RECORD OF DISCUSSIONS HELD AT THE SECOND MEETING OF THE TECHNICAL ADVISORY PANEL ON MAY 21 AND 22, 1987

AMAR GUPTA AND STUART MADNICK

The second meeting of the Technical Advisory Panel (TAP) began with presentations by eight experts on various aspects of Knowledge-Based Integrated Information Systems Engineering. These experts, drawn from government, industry, and academia, expressed their opinions in the following areas:

(a) Organization, Strategy, and Management;

(b) Distributed Database Technology;

(c) Knowledge and Semantics; and

(c) Information Modeling and Mapping.

All the experts emphasized the need for concerted action for mitigating major problems in each of these areas.

The representative of MIT described the SATORI model (derived from Strategic Applications, Technology, and Research Initiative), the Knowledge and Information Delivery Systems (KIDS) concept, and the other research activities conducted to date. He first examined the strategic aspect of how one can motivate participation in loosely-coupled organizations. Next, he described the complexities inherent in heterogeneous systems. The connectivity issue was considered in terms of physical connectivity and logical connectivity. The former deals with bandwidth, security, availability, and reliability, while the latter relates to syntax mapping, semantic mapping, and concept inferencing. The importance of establishing and adhering to focused standards was also highlighted.

The representative of Texas A&M University described the nature of the integration problem in the context of the aerospace industry, and distinguished between the product definition view, the product management view, and the product delivery view. He described the essential components of a framework for tackling the problem, and the areas characterized by major voids in the level of available methods, tools, and technologies.

Four working groups were constituted to consider the key issues identified at the end of the first meeting of TAP held in February 1987. The recommendations of these groups were discussed. The areas requiring further work, as perceived by TAP members, covered both technical and non-technical issues.

I. PARTICIPANTS

The second meeting of the Technical Advisory Panel (TAP) was held at M.I.T. on May 21 and 22, 1987. The individuals who participated were as follows:

(A) Members of Technical Advisory Panel

1.	Daniel S. Appleton	DACOM
2.	Robert R. Brown	Rand Corporation/Consultant
3.	Peter P. Chen	MIT/Louisiana State University
4.	Stu Coleman	Pacific Info Management
5.	Ugo Gagliardi	Harvard / G.S.G. Inc.
6.	John Henderson	MIT
7.	Lynette Hirschman	Unisys
8.	Frank Manola	Computer Corporation of America
9.	Mary Mitchell	National Bureau of Standards
10.	Prof. Jay Nunamaker	University of Arizona
11.	Sam Nusinow	Knowledge Base Engineering Ent.
12.	William Putnam	Rockwell International
13.	Cdr. J. Allen Sears	DARPA / ISTO
14.	Charles Standridge	A. Pritsker and Associates
15.	Paul Thompson	Control Data Corporation
16.	Gio Wiederhold	Stanford University

(C. Gordon Bell of NSF, Alfonsas F. Cardenas of UCLA, Thomas Cullinane of Northeastern University, Bud Giangrande of TSC, Ellen Knapp of Coopers and Lybrand, Raymond Liuzzi of Rome Air Development Center (RADC), John Machado of Space and Naval Warfare Systems Command (SPAWAR), Sham Navathe of IBM T.J. Watson Research Center, J. William Poduska of Stellar Computers, John F. Rockart of MIT, and Michael Stonebraker of U.C. Berkeley, could not attend this meeting.)

Complete addresses, along with telephone numbers of all experts who were invited to serve as TAP members, are enclosed as Appendix A.

(B) Representatives of the U.S. Air Force

17. Major Paul Condit
18. Mark Hoffman
19. Lt. Mike Painter

(C) Representatives of Transportation Systems Center (TSC)

20. Frank Hassler
21. Bob Berk

(D) Representatives of M.I.T.

22. Joseph Sussman
23. S. E. Madnick
24. Amar Gupta
25. N. Venkatraman
26. Y. Rich Wang
27. Subhash Bhalla
28. Ali Tobah
29. Ajit Kimbal
30. Manuel Quintana
31. Maher Kallel
32. Nitin Nohria
33. Sam Levine

(E) Representative of Texas A&M University

34. Ric Mayer

(F) Observer

35. B. Neil Snodgrass, DACOM

The meeting was organized by Massachusetts Institute of Technology, in consultation with representatives of the research sponsors (U.S. Air Force and Transportation Systems Center).

II. OBJECTIVES

For its second meeting, the objectives of the TAP were defined to be as follows:

(1) To hear the opinions of the new TAP members;
(2) To discuss the highlights of the material generated by the research organizations participating in this project;
(3) To become aware of current interests of various governmental agencies;
(4) To discuss and review the preliminary set of recommendations;
(5) To delineate a time-bound plan for progress in various areas of distributed heterogeneous database systems; and
(6) To agree on a plan for Phase II of this project.

A copy of the agenda is placed at Appendix B. During the first session held on the morning of May 21, 1987, each new TAP member presented his or her ideas in approximately 15 minutes. After each presentation, there was a question-answer session of about 5 minutes per speaker. The second session (afternoon of May 21), was comprised of presentations by representatives of MIT and Texas A&M University. At the end of this session, four working groups were constituted to study specific aspects of the problem. These working groups held their independent discussions on May 21 (evening) and May 22 (morning). The recommendations of each working group were presented by their respective chairpersons. Discussions relating to follow-up activities were held in the final session on May 22, 1987.

The highlights of each working session are described in the following sections.

III. OPENING REMARKS

The second meeting of the Technical Advisory Panel (TAP) began with opening remarks by Professor Tom Magnanti (MIT), Dr. Frank Hassler (TSC), Professor Joseph Sussman (CTS-MIT), and Major Paul Condit (Air Force). Professor S.E. Madnick and Dr. Amar Gupta, joint principal investigators for this project, emphasized the importance of the Technical Advisory Panel (TAP) and thanked the invitees for their assistance to date and for accepting the invitation to participate at this meeting.

IV. PRESENTATIONS BY INVITEES

(1) Presentation by Bill Putman

Bill Putnam presented an overview of the Integrated Design Support System (IDS) architecture. He mentioned that the B-1B project involved 300 contractors ranging from major airframe builders to small machine shop facilities. To integrate information located at the many different sites, a 3-schema architectural approach was adopted in the IDS effort. Phase I of this effort was launched in November 1984. This phase was comprised of three subphases, the last of which ended in December 1986. Currently, his company is awaiting formal approval of Phase II activities.

IDS project team effort has so far linked in excess of 2 million lines of code, using 20,000 lines of new code written specifically as part of this effort. In response to a question

about data-driven design versus prccess-driven design posed by Peter Chen, it was clarified that a data-driven approach has been utilized in IDS.

At this stage, Major Paul Condit distinguished between IISS and IDS efforts. While both projects are sponsored by the Air Force, the scope of IDS is much broader than IISS. The objective of IDS is to look at the entire life cycle, and to develop solutions accordingly.

(2) Presentation by Ugo Gagliardi

Professor Gagliardi began by stating that data and process are complementary issues, just as wave theory and particle theory are complementary. Next he emphasized that the process of reaching consensus on canonical forms for defining data types, structures, and objects is slow and cumbersome. The process begins with some people making suggestions, and the marketplace accepting a subset of them. One cannot simply make decisions and impose solutions. Canonical models and abstractions have not yet emerged in either the main area, or even in ancilliary areas such as graphics and communications.

In the communications arena, for example, there are three sets of candidates. One is the set of networking protocols promoted by Defense Advanced Research Projects Agency (DARPA). The second is Systems Network Architecture (SNA) and its clones, and the third is the one supported by DEC and the engineering community. This makes it necessary to think in terms of designing gateways between them.

Instead of transferring entire data, only the kernel of the state-of-change should be circulated. Simple data structures such as alphanumeric strings and business graphics should be considered rather than more complex ones. Rather than attempting something very ambitious and failing, it is preferable to adopt a low profile and a humility-based approach.

(3) Presentation by Frank Manola

Frank Manola described how object-oriented approaches could potentially reduce the complexity of integrating heterogeneous components. He mentioned that these approaches permitted models to be adapted to changes during the system life cycle, as well as to different applications. However, there is still no consensus on either the data model or the set of objects. Significant work is needed to define common data, and to reach agreement on data interface standards. Further, the object-oriented approach does

not take care of existing complexities. He distinguished between two sets of complexities: the complexity of abstraction, which can be reduced by choosing a different set of abstracts; and the complexity inherent in the problem, which cannot be mitigated.

Knowledge-based processing will play an important role especially in:

(i) resolving differences in data semantics;
(ii) implementing advanced concurrency control; and
(iii) implementing application-specific rules (versioning, configuration management, and security).

He concluded his presentation with a discussion of the HiPAC project, and its relevance to the present endeavor.

At this stage, there were several interesting comments. Professor Gagliardi traced the evolution of data and object types. Initially, there was Fortran language, which viewed data as being external to the program. Then came Cobol accompanied by the concept of localization of data. The latter concept subsequently led to assignment statements and print statements. Next, ADA and others promoted the view of data types. Finally, one sees the idea of data encompassing the inheritance of operations.

Professor Chen observed that there is a concept of evolution in databases too. For example, the object-oriented concept emerged from artificial intelligence. Professor Wiederhold felt that the object-oriented approach will not reduce complexities. In fact, it will increase them. However, it has to be done in order to share information.

(4) Presentation by Jay Nunamaker

Professor Nunamaker described the use of Plexsys Planning Tools, which consist of the following:

(a) Planning Tools;
(b) System Design Tools;
(c) Data Flow Diagrams;
(d) Data Dictionaries;
(e) Data Management Systems;
(f) Application Generators;
(g) System Analysers;
(h) Fourth Generation Tools; and
(i) System Implementation Tools.

Using the above set of tools in conjunction with the knowledge base, one can develop models of three different types: (i) idea generation models; (ii) information structuring and analysis models; and (iii) planning models.

He emphasized the advantages of electronic brainstorming in terms of the following:

(a) Improved quality of group dynamics;
(b) Ability to integrate ideas of different people;
(c) Higher efficiency; and
(d) Greater freedom to express opinions even when it is not palatable to bosses.

(5) Presentation by John Henderson

Professor John Henderson began his presentation by stating that minimization of risk is not always the best strategy. Zero risk usually implies zero returns.

He highlighted the fact that one cannot influence technology trends alone. As a user of computers, one does not question the objectives of computer manufacturers, as long as the user can make money from it. In general, it is necessary to redesign systems altogether after every seven years. Thus, instead of waiting indefinitely for the optimal situation to occur, it makes sense to design systems with available technology, knowing in advance that such systems will need to be discarded in due course of time.

Professor Henderson traced the interrelationships between the business domain and the the information domain, and how these domains are influenced by internal and external factors. Evolution of new technology impacts business strategy. The latter leads to organizational changes, which in turn lead to changes in strategic information systems.

(6) Presentation by Stu Coleman

Stu Coleman felt that the definition of organization is changing. Whereas it previously connoted in-house equipment and employees, the term now includes vendors and suppliers as well. In a similar vein, the definition of common data standards is also changing. He referred to the term Information Automat proposed by Max Wilson.

He emphasized that it is important to determine the order of establishing data standards and to integrate it into the business strategy of the government. Further, he felt that the assimilation process is also important.

(7) Presentation by Mary Mitchell

Mary Mitchell traced the history of activities in this area at National Bureau of Standards (NBS). They first started with SQL, then looked at the query decomposition problem, and next moved to the task of developing standards. At the presentation layer of the classification developed by International Standards Organization (ISO), they came up with ASN/1 (abbreviated from Abstract Syntax Notation/1). In addition, they also defined a neutral transport format.

She described the problem of identifying semantics of shared data resources. Since embedding of data integrity constraints cannot be done with existing technology, NBS has sponsored a project in this area at the University of Florida.

She also briefly touched upon the role of Automated Manufacturing Research Facility (AMRF), its current involvement in Product Data Exchange Standard (PDES), and why more test-beds like AMRF are needed.

(8) Presentation by Cdr. J. Allen Sears

Cdr. J. Allen Sears observed that shaping of new technology is motivated by an application focus. One is dealing not with toys, but with really large projects involving vast amounts of data.

At DARPA, there are 135 individuals, of whom 65 persons serve as program managers. Cdr. Sears described the current interests of his group, which focuses on computer technology. He elaborated on the area of intelligent knowledge/database systems. The current objectives are:

(a) to provide high-performance access to large amounts of data and knowledge;
(b) to allow knowledge-based systems to share data and knowledge among multiple users and processes; and
(c) to design and develop a new generation of knowledge/data management systems and to use them for prototype applications.

V. PRESENTATIONS BY RESEARCH TEAMS

(1) Introduction

The afternoon session began with Dr. Amar Gupta describing the role and responsibilities of various participating agencies and research organizations. He said that the funding had been provided by the Air Force. The Air Force had entered into an inter-agency agreement with the Transportaiton Systems Center (TSC), which in turn made use of an existing basic ordering agreement between TSC and the Center for Transportation Studies at MIT. The actual research work had been conducted by the staff of Sloan School of Management at MIT. In addition, MIT had appointed Texas A&M University as a subcontractor, which in turn benefitted from the services of Tom Cullinane of Northeastern University, Stu Coleman of Pacific Information Management, and Sam Nusinow of Knowledge Base Engineering Enterprises. The work performed at MIT and other agencies complemented each other: MIT focused on strategic issues, while Texas A&M University focused on tactical issues; MIT concentrated on database aspects, while Texas A&M University looked at system design issues; and MIT had attempted to come up with generalized solutions to the problem, while Texas A&M University had tried to develop specialized solutions for the Air Force environment.

(2) Presentation by MIT Team

Next, Professor Stuart E. Madnick elaborated on the work performed so far at MIT. He described the SATORI model (derived from Strategic Applications, Technology and Research Initiative), the Knowledge and Information Delivery System (KIDS) concept, and the various dimensions to the problem. He mentioned that 21 technical reports had been generated by the MIT research team so far, with each report focusing on a specific aspect of the problem. (The list of technical reports is placed as Appendix C.)

The term "Integrated Information Systems" is used to connote integration at several different levels. Integrated Systems can span:

(a) Applications, such as integration of CAD/CAM and documentation;
(b) Functional areas, such as procurement, engineering, and logistics;
(c) Organizational boundaries, such as integration of project status from Rockwell, TRW, and Lockheed; and
(d) Geographically dispersed units around the globe.

He first examined the strategic aspect of how one can motivate participation in loosely-coupled organizations. The concept of inter-dependent value chains was described using examples of a major regional hospital and a large international bank.

The discussion then turned to technical issues, beginning with the properties of currently-available distributed homogeneous database systems such as Oracle and Ingres. The following six properties were proposed:

(a) Retrieval Transparency;
(b) Update Transparency;
(c) Schema Transparency;
(d) Performance Transparency;
(e) Transaction Transparency; and
(f) Copy Transparency.

Next, the additional complexities inherent in heterogeneous systems were mentioned. Such systems are characterized by multiple data models (DDL) and multiple query languages (DML). Eight different prototype systems were compared and contrasted.

Professor Madnick decomposed the connectivity issue into physical connectivity and logical connectivity. The former deals with issues of bandwidth, security, availability and reliability. The latter deals with syntax mapping, semantic mapping, and concept inferencing. Concept inferencing was discussed in terms of two prototype systems called KOREL (Knowledge Oriented Representation Language) and ADBMS (Abstract DBMS). The issues of syntax mapping and semantic mapping were discussed using a number of examples drawn from diverse fields. In addition, the importance of establishing and adhering to standards was highlighted. Overall, it was felt that the main problem could be separated into several sub-problems in the following areas:

(a) Organizational and Strategic Issues;
(b) Distributed Database Technology;
(c) Knowledge and Semantics; and
(d) Information Modeling and Mapping.

The above issues are examined in detail in Technical Reports 4 - 22.

(3) Presentation by Texas A&M University Team

Richard J. Mayer presented the highlights of the work performed at Texas A&M University and at various subcontractor sites. He described the nature of the integration problem in the context of the aerospace industry, and distinguished between the product definition view, the product management view, and the product delivery view. He described the essential components of a framework for tackling the problem, and the areas characterized by major voids in the level of available methods, tools, and technologies. In the area of methods, he mentioned that there was lack of formalization, as well as the need for education and technology transfer. The area of tools was considered in terms of component tools and prototype tools. In terms of the former, the needs are for:

(a) Modeling Tool Generation Utilities;
(b) Tool Data Exchange Standards; and
(c) Seed Funding for Tool Integration.

Prototype tools are needed for:

(a) Data State Modeling
(b) Process Modelling;
(c) Systems Architecture Capture and Simulation;
(d) Coding, Classification and Cluster Analysis;
(e) Conceptual Schema Design Generation from Information Models;
(f) Rapid Prototyping of External Schema;
(g) Microcomputer-based Data Collection; and
(h) Rapid Prototyping of IDS Applications

The above deficiencies in tools must be addressed as a part of the structured framework.

Mr. Mayer discussed his estimate of the amount of time needed to fill existing voids in methods, tools, and technolo-gies. His presentation concluded with the following list of thrust areas:

(a) Framework Development;
(b) Component Methods;
(c) Technology Transfer;
(d) Environment;
(e) Component Tools; and
(f) Knowledge Based Tools.

Further discussion of all these areas is contained in Technical Report #3.

(4) Constitution of Working Groups

After the above presentations, four working groups were constituted to discuss specific aspects of the problem and to formulate a concrete set of recommendations in the respective areas. The topics assigned to the working groups were based on the consensus reached at the first meeting of TAP held in February, 1987. A list of members of each of these working groups, and their respective chairpersons, is placed as Appendix D. These four working groups met during the evening of May 21 and the morning of May 22. The recommendations of the different working groups are described in the next section.

VI. RECOMMENDATIONS OF WORKING GROUPS

The recommendations of the four working groups were as summarized below.

(1) Working Group #1: Knowledge and Semantics
Chairperson: Gio Wiederhold

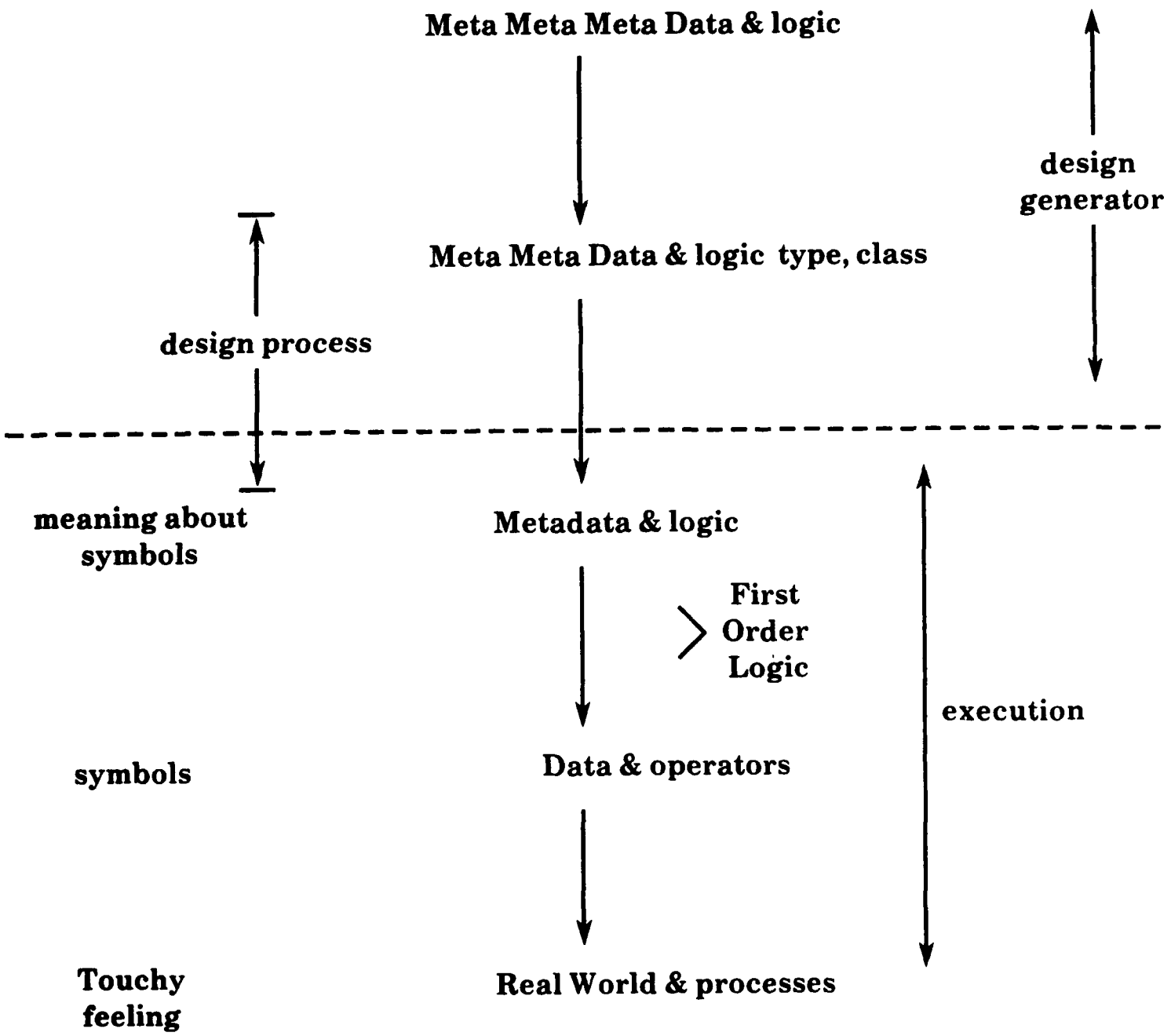
On behalf of his working group, Professor Wiederhold emphasized the multiple hierarchical aspects involved in dealing with knowledge issues. Some of these aspects can be represented as depicted in the diagram shown on the next page.

In view of the reality which the diagram illustrates, it is necessary to conduct further research in many areas. The working group proposed the following list of priority areas:

(a) To formalize the intuitions behind, and the experience with, existing methods;

(b) To quantify effectiveness of existing methods, and to close the big feedback loop; and

(c) To investigate emerging formalisms for representing natural, real world semantics and knowledge, for use in representing information system semantics (complements existing research in Logic Extensions).


Meta Meta Meta Data & logic
design generator
Meta Meta Data & logic type, class
design process
meaning about symbols
Metadata & logic
First Order Logic
execution
symbols
Data & operators
Touchy feeling
Real World & processes

(2) Working Group #2: Evolution of Information Model
Chairperson: Sam Nusinow

Speaking on behalf of his working group, Mr. Nusinow began by observing that in the case of both IDEF0 and IDEF1, the concepts are neither fully understood nor fully documented. In fact, they are probably not even fully correct. Further, IDEF1 does not permit easy incorporation of temporal changes. In view of these problems, it is necessary to undertake significant effort in several areas. In particular, further work is needed in the following aspects:

(a) Model Integration

- Inter model (IDEF 0/1/2)
- Intra model (Views)
  -- External Views
  -- Conceptual Views
  -- Co-Existence of Conceptual Views
- Mapping

(b) Knowledge Modeling

- Frames
- Object Oriented

(c) Levels of Abstraction

- Strategic, Tactical, Operational
- Decomposition
- Aggregation

(d) Model Extensions (See Section 3.2.3 of Technical Report #3)

- State transition rules
- Complex Constraint Specification
- Temporal Dimension
- Flow Timing

(e) Multiple Model Framework

- Requirements Rationalization
- Identification of Methods and Voids (Data State Model, etc.)
- What is possible versus what is practical

(f) Model Life Cycles

- Delivery System Network
- Data Semantics
- Inference Logic
- Application
- Strategic, Tactical, Operational
- Large/Small

(g) Model Application

(h) Model Translation

- IDEF1 ---> Object Oriented

(3) Working Group #3: Performance and Feasibility of Distributed Databases
Chairperson: Mary Mitchell

On behalf of her group, Mary Mitchell observed that while partial solutions exist for special cases, sustained effort is required to come up with effective solutions for generalized applications. Various issues were presented, a first qualitative estimate was made of the level of importance and the level of complexity in each issue, and the present status of the issue was described. The consensus was as summarized in the following table.

ISSUES	CAN BE (relates to system which can be impiemented with current technology)	NOW (relates to system as they currently exist)	LEVEL OF RESEARCH (has gone on, or is proceeding currently)
(1) Correctness			
(a) Concurrency Control	High importance; Hard;	Hard;	Lot; Some;
(b) Validity Constraint	Low importance; Hard;		
(2) Global Updates	High/Medium importance, Low difficulty in supporting for special cases;	Low importance; Hard;	Little;
(3) Optimization	High importance, Hard (Application dependent);	Hard; Low demand;	Some;
(4) Neutral Operations (Hook to Legacy)	High importance, Low difficulty (Brute Force);	High difficulty;	Little;
(5) Security	High importance; Hard;	Low importance; Low demand;	Little outside of DOD;
(6) Reliability and Recovery	High importance; Hard;	Hard;	Some;
(7) Replicated Data	High importance (Given performance and reliability, low difficulty given other);	Low importance; Hard;	Little;
(8) Autonomy	High importance; Low difficulty;		Little;
(9) Management	High importance; Hard;	Low difficulty;	None;

(4) Working Group #4: Organization, Strategy, and Management
Chairperson: John Henderson

This working group made the following valuable suggestions:

(a) Choose a problem of relevance as a research domain

- Logistics Management Systems
- Very Large Project Management

(b) Focus on technologies and Methodologies that will change the way individuals or groups interact to plan, design, implement, etc.

- Electronic Networking
- Multi Media
- Integrated Group Process Methodology

(c) Focus on technologies and strategies to manage and coordinate inter-organizational processes

- Multiple Agents
- Impacts on Social Networks

(d) Focus on strategies to position and align the organization to exploit existing and emerging innovation

- Methodological
- Technical

(e) Focus on Strategies that will enable/catalyze major organizational infrastructure transformation

- Technology
- Methodology

(f) Develop ways to effectively identify/integrate emerging skills and specializations

VII. Critical Issues

After the chairpersons of the four working groups had made their presentations, each attendee was requested to specify the issue he or she felt was most critical in the present context. All attendees were also requested to make an estimate of the level of effort suggested, in terms of person years, to overcome the most important problems. The opinions are summarized below:

NAME	MOST IMPORTANT ISSUE	LEVEL OF EFFORT SUGGESTED
(a) Appleton	Define requirements for three schema system development methodology that can be applied on IDS.	Not indicated
(b) Brown	Identifying, evaluating and selecting a single language for representing all levels of KBIISE.	5 person-years
(c) Chen	Unified model for data/ knowledge.	5-10 person-years
(d) Coleman	Methods for an organization to develop a workable insertion strategy and define the assimilation mechanisms for IDS-like technological capabilities.	7-10 person-years
(e) Hassler	Development of a general systems (management) science as a context for effective and efficient KBIISE.	Not indicated
(f) Henderson	Study how technology will change the way people work together.	3-5 persons for 3 years each
(g) Hirschman	Tools for building coherent, expressive models for information systems applications.	4-6 person-years

(h) Manola	Integration of processes and data/knowledge (in an efficient way).	1-2 years for initial cut
(i) Mayer	Formulation of the methods of the system development process and the construction of an integrated support environment based on the resulting structures.	35 person-years for operational system; 14-15 person-years for demonstratable prototype
(j) Mitchell	Techniques for making the semantics of data and assumptions accessible.	Not indicated
(k) Nusinow	Need for a complete integrated information system evolution environment.	150-250 person-years
(l) Putnam	Framework for developing large integrated information and knowledge based systems.	70 person-years for initial effort
(m) Snodgrass	A definition of the concepts, infrastructures, and potential technologies for transitioning to inter-organizational integration.	1-3 person-years
(n) Standridge	Understanding of the operational dynamics, resource requirements, and performance characteristics of large-scale, heterogeneous DBMS.	2 person-years
(o) Sussman	Data systems in support of new organizational form.	Not indicated
(p) Th[illegible]pson	Development of Multi-model Framework.	Between 0.5 person-years and infinity

(q) Venkatraman	Understanding the motivation and incentives necessary for managing "interorganizational" processes.	2-5 person years
(r) Wiederhold	(i) Closing the design to methodology feedback loop; and (ii) Semantics of uncertain and time-based variables.	(i) 10-50 person years; and (ii) Not indicated

There was general consensus that it would be necessary to address the above issues, at least to some degree, in order to come up with an efficient unified methodology that will enable integration of large databases, both from a technological perspective as well as from an organizational perspective. Everyone agreed that the problems created by multiple non-integrated databases are becoming increasingly common, and there was an urgent need to carry out concerted efforts to mitigate these problems. The key areas requiring further work, as perceived by TAP members, are summarized in the following table.

Name	Knowledge and Semantics	Information Models	Distributed Databases	Organization and Strategy
(a) Appleton	X	X	X	
(b) Brown	X	X	X	
(c) Chen	X	X	X	
(d) Coleman		X		X
(e) Hassler				X
(f) Henderson				X
(g) Hirschman		X		
(h) Manola	X		X	
(i) Mayer		X		
(j) Mitchell	X		X	
(k) Nusinow		X		X
(l) Putnam	X	X	X	
(m) Snodgrass		X		
(n) Standridge			X	
(o) Sussman	X	X		X
(p) Thompson	X	X		
(q) Venkatraman				X
(r) Wiederhold	X		X	

VIII. Conclusion

In his concluding remarks, Dr. Gupta thanked the invitees for participating in this meeting. He felt that there was a great interest in the objective of this project, and that there appeared to be enough support for convening meetings of TAP on a regular basis.

Dr. Gupta requested all the participants to submit technical papers elaborating on their respective views. The intention was to bring out an edited book, which would be published and distributed by a professional organization such as IEEE. He requested the participants to send in their papers as soon as possible. This would enable the research team to integrate the opinions in the final report which was intended to be completed during June and July, 1987. Dr. Gupta also mentioned that any comments and suggestions relating to the technical documents generated by MIT and Texas A&M University would be most welcome.

Speaking on behalf of the sponsors, Major Condit thanked all the attendees for their participation and their valuable inputs. He added that the level of effectiveness cannot be judged by the level of agreement or disagreement. It is the level of effectiveness that really counts. He requested all the TAP members to use as many examples as possible in their respective papers. Such examples enable the readers to grasp more readily the viewpoint of the writer(s).

Based on the research work performed by MIT, Texas A&M University, and other subcontractors and members of the TAP, the project team at MIT will come up with a final report. This report will contain recommendations as well as suggestions for Phase II of this project. It was agreed that MIT would act as the clearing house for receiving and circulating ideas concerning this effort.

APPENDIX A

LIST OF INVITEES FOR THE MEETING OF THE TECHNICAL ADVISORY PANEL HELD AT M.I.T. ON MAY 21 AND 22, 1987

Daniel S. Appleton, President
D. Appleton Co., DACOM
1334 Park View Avenue, Suite 220
Manhattan Beach, CA 90266
(213) 546-7575

C. Gordon Bell
Assistant Director, CISE-NSF
1800 G. Street, N.W.
Washington, D.C.
(202) 357-7373

Howard Bloom (Alternate for Mary Mitchell)
Chief, Factory Automation Systems Div.
National Bureau of Standards
Building 220, Room A-127
Gaithersburg, MD 20899
(301) 975-3509

Robert Brown
2229 Via Caliente
Fullerton, CA 92633
(714) 773-4050

Alfonsas F. Cardenas
Dept. of Computer Science, UCLA
3731 Boelter Hall
Los Angeles, CA 90024
(213) 825-7550

Peter P. Chen
Department of Computer Science
Louisiana State University
Baton Rouge, LA 70808
(504) 388-2482

Stu Coleman
Pacific Information Management, Inc.
2121 Cloverfield Blvd., Suite 203
Santa Monica, CA 90404
(213) 829-3380

Thomas Cullinane
309 Snell Engineering Building
Northeastern University
360 Huntington Ave.
Boston, MA 02115
(617) 437-4856

Ugo Gagliardi
c/o G.S.G. Inc.
51 Main Street
Salem, N.H. 03039
(603) 893-1000

John C. Henderson
MIT - Sloan School of Management
E53-313
Cambridge, MA 02139
(617) 253-4319

Lynnette Hirschman
Manager, Logic Based Systems R & D
Unisys
P.O. Box 517
Paoli, PA 19301
(215) 648-7554

Ellen Knapp
Coopers and Lybrand
1800 M Street N.W.
Washington, D.C. 20036
(202) 822-4000

Raymond Liuzzi
RADC/COTD
Building 3
Griffis AFB,
Rome, NY 13441
(315) 330-2643

John Machado
Space and Naval Warfare Systems Command
Code 613
2511 Jefferson Davis Hwy.
Arlington, VA 22202
cc: Norma Stopyra

Frank Manola
Computer Corporation of America
4 Cambridge Center
Cambridge, MA 02142
(617) 492-8860

Ric Mayer
Director, Knowledge Based Systems Lab
Room 143
Engineering Research Center
Texas A&M University
College Station, TX 77843
(409) 845-9363

Mary Mitchell
National Bureau of Standards
Building 220, Room A-327
Gaithersburg, MD 20899
(301) 975-3538

Sham Navathe
IBM T.J. Watson Research Center, Hawthorne
30 Saw Mill River Road (Route 9A)
Room No. h3-C04
P.O. Box 218
Yorktown Heights, NY 10598
(914) 789-7085

Jay F. Nunamaker
MIS Dept., College of BPA
University of Arizona
Tucson, AZ 85721

E.I. (Sam) Nusinow
Knowledge Base Engineering Enterprises
240 Elmwood Drive
Centerville, OH 45459
(513) 434-1716

J. William Poduska
Chairman, Stellar Computers
100 Wells Ave.
Newton, MA 02159
(617) 964-0288

William Putnam
Rockwell International
North American Aircraft Division
100 N. Sepulveda Blvd.
El Segundo, CA 90245
(213) 647-3730

John F. Rockart
MIT Sloan School of Management
E 40 - 187
Cambridge, MA 02139
(617) 253-6608

Cdr. J. Allen Sears
Program Manager, DARPA/ISTO
1400 Wilson Blvd.
Arlington, VA 22209-2308

Charles R. Standridge
A. Pritsker and Associates
1305 Cumberland
P.O. Box 2413
W. Lafayette, IN 47906
(806) 794-7696

Michael Stonebraker
Department of Computer Science
U.C. Berkeley
571 Evans Hall
Berkeley, CA 94702
(415) 642-5799 (or 1024)

Paul Thompson
Control Data Corporation
HQB01D
8100 34th Avenue So.
Bloomington, MN 55420
(612) 853-6062

Gio Wiederhold
Stanford University
Margaret Jacks Hall, Room 436
Palo Alto, CA 94305
(415) 723-0685

APPENDIX B

KNOWLEDGE-BASED INTEGRATED INFORMATION SYSTEMS ENGINEERING PROJECT

AGENDA FOR
SECOND TAP MEETING TO BE HELD ON MAY 21 AND 22, 1987
AT MIT IN THE SENIOR EXECUTIVES ROOM, (E52-175),
50 MEMORIAL DRIVE, CAMBRIDGE, MA 02139

Day 1: May 21, 1987

8:45 to 9:00 — **Coffee and Informal Discussions**

9:00 to 9:20 — **Opening Remarks**
- Tom Magnanti, MIT
- Frank Hassler, TSC
- Joseph Sussman, CTS-MIT
- Major Paul Condit, Air Force
- S.E. Madnick / Amar Gupta, MIT

9:20 to 10:20 — **Approaches to Problem**
- (i) Bill Putnam - The IDS Approach
- (ii) Ugo Gagliardi - Major Issues in Distributed Databases
- (iii) Mary Mitchell - Recent Developments on NBS Sponsored Efforts

10:20 to 10:30 — **Coffee Break**

10:30 to 12:00 — **Approaches to Problem (continued)**
- (iv) Frank Manola - Research in Distributed Databases
- (v) Jay Nunamaker - Issues and Alternatives
- (vi) John Henderson - Interorganizational Aspects and Policies
- (vii) Stu Coleman - Related Issues

12:00 to 12:20 — **Efforts Sponsored by Government**
- Cdr. J. Allen Sears - Research efforts sponsored by DARPA

12:20 to 1:20 — **Lunch in Faculty Club**

1:20 to 3:00	**Presentations by MIT Research Team** **Stuart E. Madnick / Amar Gupta**
3:00 to 3:10	**Coffee Break**
3:10 to 4:40	**Presentations by Subcontractors** **Rick Mayer**
4:40 to 5:40	**Constitution of Working Groups and Presentation of Draft Recommendations** **Stuart E. Madnick / Amar Gupta**
5:40 to 7:00	**Drinks and Dinner in Schell Room**
7:00 to 8:30	**Meetings of Working Groups**
8:30 to 9:00	**Meeting of chairpersons of working groups**

DAY 2: MAY 22, 1987

9:00 to 9:15	**Coffee and Informal Discussions**
9:15 to 10:00	**Private meetings of each Working Group**
10:00 to 10:20	**Presentation by chairperson of WG-1 and Discussion**
10:20 to 10:40	**Presentation by chairperson of WG-2 and Discussion**
10:40 to 10:50	**Coffee Break**
10:50 to 11:10	**Presentation by chairperson of WG-3 and Discussion**
11:10 to 11:30	**Presentation by chairperson of WG-4 and Discussion**
11:30 to 12:00	**Discussion of Recommendations and Future Efforts**
12:00 to 1:00	**Lunch in Faculty Club**
1:00 to 2:00	**Discussion**

APPENDIX C

KNOWLEDGE-BASED INTEGRATED INFORMATION SYSTEMS ENGINEERING PROJECT

Exploratory Research Efforts and List of Technical Reports

(as of May 21, 1987)

Principal Investigators: S.E. Madnick and Amar Gupta, M.I.T.

NUMBER	TITLE OF REPORT	PRIMARY RESPONSIBILITY
1.	*EXECUTIVE SUMMARY*	AMAR GUPTA AND S.E. MADNICK
2.	RECORD OF DISCUSSIONS HELD AT TAP MEETINGS ON FEBRUARY 17, 1987	AMAR GUPTA
3.	CONSOLIDATED REPORT SUBMITTED BY TEXAS A&M UNIVERSITY AND OTHER SUBCONTRACTORS	TEXAS A&M
4.	EVOLUTION TOWARDS STRATEGIC APPLICATIONS OF VERY LARGE DATA BASES THROUGH COMPOSITE INFORMATION SYSTEMS	S.E. MADNICK / RICH WANG
5.	A TECHNICAL COMPARISON OF DISTRIBUTED HETEROGENEOUS DATABASE MANAGEMENT SYSTEMS	SUBHASH BHALLA / B.E. PRASAD
6.	A CONCEPTUAL MODEL FOR INTEGRATING AUTONOMOUS PROCESSING	BILL FRANK / S.E. MADNICK / RICH WANG
7.	STANDARDS FOR DATA EXCHANGE IN AN INTEGRATED ENVIRONMENT: A METHODOLOGICAL APPROACH	MAHER KALLEL
8.	INFORMATION TECHNOLOGIES AND INTER-ORGANIZATIONAL NETWORKS: STRATEGIC AND ORGANIZATIONAL IMPLICATIONS	NITIN NOHRIA / N.VENKATRAMAN
9.	INTERFACING OBJECTS AND DATABASES	SAMUEL P. LEVINE

10.	AN EXPERT SYSTEM FOR ACCESSING AND INTEGRATING DESIGN ANALYSIS KNOWLEDGE	WILLIAM H.B. HABECK
11	DISTRIBUTED DATABASE SYSTEMS: A COMPARISON BETWEEN ORACLE AND INGRES	BOB GREF
12.	GAINING STRATEGIC ADVANTAGE THROUGH COMPOSITE INFORMATION SYSTEMS	JOSEPH LOUIS MASSIMO
13.	EXPERT SYSTEMS FOR RESOLVING SEMANTICS BETWEEN DATABASES	MARINA POCATERRA
14.	CASE STUDY OF INTEGRATED AUTONOMY	MARIA RINCON
15.	DATA COMMUNICATION NETWORKS: A COMPARATIVE EVALUATION OF THE MIT AND HARVARD ENVIRONMENTS	PHILLIP S. YOO
16.	ACHIEVING A SINGLE MODEL FOR INTEGRATING HETEROGENEOUS DATABASES	DANIEL KENNEDY
17.	KNOWLEDGE-BASED PICTORIAL INFORMATION SYSTEMS	GEORGE APOSTOL
18.	STORAGE AND RETRIEVAL OF PICTORIAL INFORMATION IN HETEROGENEOUS COMPUTING SYSTEMS	LOWELL W. KIM
19.	THE USE OF STANDARD DATA DEFINITIONS IN COMPOSITE INFORMATION SYSTEMS	ANDREW TRICE
20.	AN ACTOR'S ROLE IN INTEGRATING EXPERT AND DATABASE SYSTEMS	JAY COOPRIDER
21.	ANNOTATED BIBLIOGRAPHY	ALI TOBAH
22.	TOWARDS A CIS MODEL FOR STRATEGIC APPLICATIONS	CHARLEY OSBORN

LIST OF WORKING GROUPS

WG 1:
Knowledge and Semantics

- Bob Berk
- Nick Bernstein
- Bob Brown
- Peter Chen
- Stu Coleman
- Lynnette Hirschman
- Rich Wang
- Gio Wiederhold, Chairperson
- Ali Tobah

WG 2:
Evolution of Information Model

- Dan Appleton
- Bud Giangrande
- Amar Gupta
- Mark Hoffman
- John Machado
- Richard J. Mayer
- Sam Nusinow, Chairperson
- Paul Thompson
- Maher Kallel

WG 3:
Performance Feasibility of Distributed Databases

- Subhash Bhalla
- Stuart E. Madnick
- Frank Manola
- Mary Mitchell, Chairperson
- Mike Painter
- Bill Poduska
- Bill Putnam
- Cdr. J. Allen Sears
- Charles Standridge
- Manuel Quintana

WG 4:
Organization, Strategy, and Management

- Major Paul Condit
- Ugo Gagliardi, Chairperson
- Frank Hassler
- John Henderson
- Ellen Knapp
- Jay Nunamaker
- Joseph Sussman
- Norma Stopyra
- N. Venkatraman
- Nitin Nohria

CONTRIBUTIONS BY MEMBERS OF THE TECHNICAL ADVISORY PANEL (TAP)

Members of the Technical Advisory Panel (TAP) have contributed ten excellent papers on various aspects of Knowledge-Based Integrated Information Systems Engineering. The titles of these papers, the names of the respective TAP members, and the contents of the papers are as summarized below.

(a) DISTRIBUTED DATABASES AND KNOWLEDGE BASES

(i) *KSYS: An Architecture for Integrating Databases and Knowledge Bases* (Gio Wiederhold): Describes a project that attempts to elucidate and demonstrate methods to deal with the encoding and the interpretation of conceptual knowledge and conceptual data.

(ii) *Applications of Object-Oriented Database Technology in Knowledge-Based Integrated Information Systems* (Frank Manola): Describes several applications of object-oriented database technology in knowledge-based integrated information systems, including the integration of heterogeneous system components and heterogeneous data and knowledge representations.

(iii) *A Distributed Database Architecture for an Integrated Manufacturing Facility* (Mary Mitchell): Presents the architecture of an Integrated Manufacturing Data Administration System (IMDAS) which provides the users and the application software with a unified view of the global database that is physically distributed across dissimilar component systems.

(b) INFORMATION MODELING

(i) *Reference Models for Information Engineering* (Paul Thompson): Discusses three reference models followed by a proposed engineering framework for data-driven conceptualization, specification, design, and development.

(ii) *Entity Link Key Attribute Semantic Information Modeling* (Robert R. Brown): Describes an information modeling technique which is capable of capturing an understanding of the information with which a system deals.

(iii) *Description of a Semantic Data Engineering and Product Data Configuration Control System Environment* (E.I. Sam Nusinow): Discusses the need for a Semantic Data Engineering Tool that can be used to create and to maintain a common data knowledge base.

(c) RELATED ISSUES

(i) *Natural Language Interfaces for Large-Scale Information* (Lynette Hirschman): Outlines the role and availability of natural language processing in managing large scale information processing applications and the advent of intelligent machine assistants.

(ii) *Computer-Aided Support of Deliberation and Judgment: A Process-Oriented Approach to DSS Design* (J.F. Nunamaker): Describes design, implementation, and evaluation of an automated system to support complex, unstructured group decision processes within organizations.

(iii) *Thoughts on Information Asset Management* (Daniel S. Appleton): Emphasized the need for conscious endeavor for efficient evolution and management of information resources.

(iv) *A Modeling Support Laboratory for Large Scale, Distributed, Heterogeneous Information Systems Design* (Charles R. Standridge): Describes integration of a simulation language, a simulation support system, and application related material in the context of distributed, heterogeneous information systems.

KSYS: AN ARCHITECTURE FOR INTEGRATING DATABASES AND KNOWLEDGE BASES

Gio C.M. Wiederhold
Michael G. Walker
Waqar Hasan
Surajit Chaudhuri
Arun Swami
Sang K. Cha
Xiaolie Qian
Marianne Winslett
Peter K. Rathman
Linda DeMichiel

Stanford University

Contributed by Gio Wiederhold

1. Introduction

As systems using artificial intelligence grow in size and importance, it behooves us to consider the data and software engineering required to implement substantial such systems. We define here *data engineering* as the approach which focuses on the structure of the data, while *software engineering* focuses on the computing procedures. In artificial intelligence (AI) systems, these two aspects of system design are characterized on one hand by a conce n about the representation of knowledge and on the other hand by the capabilities of the the interpreters which operate on the knowledge[1].

Knowledge representation is recognized as a major concern in AI systems. The encoding of knowledge is difficult because of the intrinsic complexity of knowledge. This complexity is evidenced by the many and varied relationships we can define among the conceptual units needed to represent knowledge. On the other hand, structures intended to deal with large quantities of factual data tend to be simple and regular; and much of the recent progress in dealing with large databases is due to the structural simplification provided by the relational model[2].

In order to deal with demands as posed by growing AI systems, a substantial research effort is now devoted to introducing more flexible support structures into databases, as indicated by the collections in[3, 4]. Many researchers in the intersection of AI and databases expect that database techniques will provide the necessary assistance to deal with the problems encountered as knowledge bases grow[5], although this belief is far from universal[4]. No convincing demonstration of the benefits of such an integration can yet be shown.

1.1. Motivation

Why should problems arise with large knowledge bases? We see first of all that the number of candidate relationships grows rapidly with the number of nodes in the knowledge base. Nodes in knowledge-bases represent concepts that are linked to many other concepts and instances in the knowledge-base. Those concepts represent abstractions at a variety of levels and classifications. Many conceptual nodes also apply to many instances. The nodes we classify as being conceptual are linked to many other concepts, while the nodes classified as being factual need only be hierarchically linked to their descriptor nodes. These descriptors define the entity being described and the attribute defining the describing value.

Traditional databases, being concerned with facts, can use relational tables effectively for the representation of their contents. The relational table provides a regular linkage structure in two dimensions, namely to the attribute and to the entity represented by the key. Maintenance of such a database is fairly straightforward[6]. In time-oriented databases a third dimension, time, is added, and this seemingly simple extrapolation, although long recognized as being useful[7], is just now being dealt with formally[8]. Things get yet worse when more structural relationships are to be considered, such as derived views which join multiple database structures [9].

Updating of the arbitrary structures used in AI is more difficult yet, in general it is probably n.p. hard as discussed by[10] for rule-based representations. Knowledge base maintenance is rarely addressed in current systems, primarily because so few knowledge-based systems of significant size have moved from the lab into practical use. Thus, issues of updating and consistency that are familiar to database managers still require solutions in knowledge base applications. Changes in the knowledge base require considerable retesting. Even then correctness is rarely provable in substantial systems.

In the discussion above we have casually introduced our definition of what is knowledge versus data. We will restate this definition now, since it is the basis for one of the two strategies we are trying to exploit in the research described in this paper. We define *data* as the factual, observable and verifiable descriptions of real-world events, and *knowledge* as the higher level concepts used to structure, classify, and relate such real world events. When the two are brought together, we can produce information[11]. A system which applies encoded knowledge to formalized collections of data, or databases, is defined to be a *knowledge-base*

system.

These definitions have to be interpreted with some practical engineering sense. For instance, the data describing the contents of a bank account are not visually verifiable: there is no pile of money in the bank with a name on it, and yet, this information is certainly best classified as data. Also, records of events which occurred in the past are no longer verifiable, but are still considered to be data. Some other examples will be harder to classify. For instance, derived data are not dealt with in this classification. This issue is not discussed in this paper, although it is of concern to us.

1.2. Our Approach

Our Knowledge SYStem project (KSYS) is intended to demonstrate methods towards a systematic approach to deal with large knowledge bases. Given the semantic distinction we make between knowledge and data, we make a corresponding engineering distinction. The notions underlying our approach are largely based on a variety of previous research in the KBMS and RX groups, initiated in 1977. Projects related to these data engineering issues of KSYS were described in[12, 13, 14, 15, 16, 17, 18]. An overview of earlier KBMS work is provided in [19].

First of all, we will use in KSYS simple, regular structures for data, while supporting the complexity of the stored knowledge with a more general structure. The benefits expected from this dichotomy are that now the costly maintenance of complex structures is only required for a fraction of the stored information, and that well-established, efficient algorithms can be applied to the database. This benefit is only significant if the volume of information that can be classified as data is large. Our understanding of the few large knowledge-bases now in existence, such as R1/XCON[20] is that this is indeed the case. We believe in fact that for all large AI systems the amount of complex stored knowledge, by our definition, is quite modest, especially in systems which require regular updating.

There are obviously also liabilities associated with the use of two distinct representations. The processing system has to provide an interface so that for a user needing information from the system, the knowledge and data are tightly coupled.

Since the user will always access the data via the knowledge-base, we believe we will find solutions to the user interface issue. In current database management systems, all accesses are already mediated by the a database schema. The traditional schema focuses on data attributes, their representation, and on access paths. Stored knowledge in the knowledge base can be viewed as a generalization of such a schema. The extension to the schema in a knowledge-based system can encompass concepts not directly represented in the database, for example overall health-status in a medical record. The knowledge base will also permit effective storage of interrelational constraints and processing rules.

The interface must also not reduce performance excessively, so that the benefits obtained from the specialized storage management can overcome this potential liability. We are not addressing in KSYS the problems of persistent storage of the conceptual frames within a DBMS. This is a real issue, and is being addressed by many researchers[21] <<Editor: Anything in this TSE issue?>>]. We expect these efforts to be successful and solve our problem.

The problem of operational efficiency in access is being addressed by keeping the knowledge in (real) memory. We assume that the memory capacity of modern computers is adequate to hold all of the required knowledge once the more voluminous data are separated. For transactions where accessed data is reused, such data can be cached, and also retained in memory. For best use, the data, once retrieved, should be bound into the knowledge structure, as explored in[22]. (See also[23] and the section below on binding and query optimization).

We now come to the second of the two strategies for knowledge management: the use of multiple hierarchies. Although computer memory hardware is now adequate to store large amounts of knowledge, this does not address the problem of how to manage that knowledge. The complexity of interactions among the knowledge grows rapidly with the size of the knowledge, eventually overwhelming the access advantage gained from the use of a large memory. Much of this complexity of interaction can be mitigated through the use of

hierarchical organization, in which all interactions flow through well defined parent-child links. However, a single hierarchical organization is an inadequate paradigm for the representation of knowledge bases, especially those which involve sharing of knowledge.

1.3. Sharing

A large system, and hence the representations used to demonstrate KSYS, must support a diversity of usage. It is desirable to share knowledge as well as data, since knowledge acquisition is even more costly than data collection[24]. Sharing of knowledge is also motivated by the objective of consistency. We expect many applications to work on different aspects of the problem. Databases technology has enabled data sharing, so that factual observations used by distinct applications will be identical. But if these applications interpret the same facts differently, because of differences in their encoded knowledge, we will not have achieved the objective of consistency among applications.

To enable sharing we see multiple hierarchical classifications of knowledge structures as a solution. Specific AI and database systems, which use general processing strategies, most often are constrained to operate in hierarchical structures, and have great difficulty in dealing with cycles. These hierarchies are closely related to the concepts of views, as further discussed below.

1.4. KSYS Research Projects

In the following sections we describe research projects within the KSYS project that are related to the theme of artificial intelligence and data engineering. Each project addresses an aspect of the theme such as simplifying the design, implementation, or maintenance of databases, knowledge bases, or programs, or providing user interfaces that reduce the need for programmer involvement.

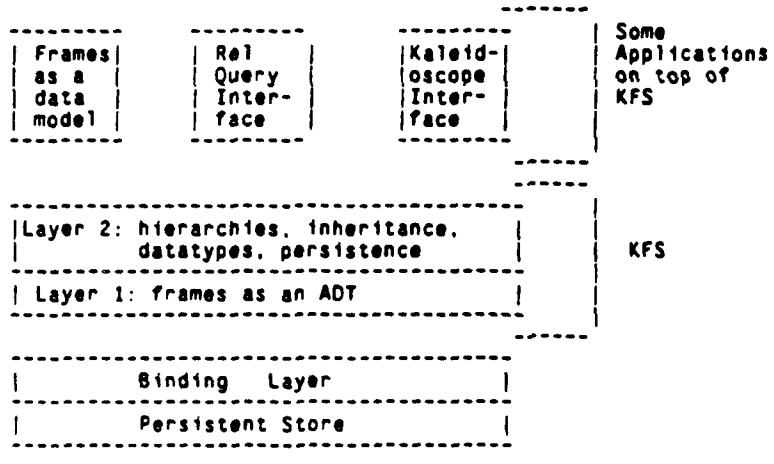


Fig 1: KBMS Architecture

The overall KSYS architecture is shown in Fig 1. The persistent store is a relational database (RDB from DEC in our prototype). The KSYS Frame System (KFS) is described in the first section following the introduction. The binding layer seeks to make optimal use of a moderately large main memory (8 megabytes), and is described in the section below on binding and optimization. Components of KSYS that use knowledge for dealing with views, semantic mediation of operations across disparate domains, integrity control and constraint management, and user interfaces are described in succeeding sections, and we conclude with work on knowledge acquisition from databases.

2. The KSYS Frame System

The KSYS Frame System (KFS) is intended to be part of an experimental Knowledge Base Management System (KBMS). The KFS will provide a base on which "intelligent" database

applications may be built. We plan to use the frame system with connections to a persistent store as the bottom layer for supporting such applications. Frames were proposed by Minsky[25] as part of a scheme for making intelligent machines. Commercial frame-based systems like KEE[26] are widely available and familiar to most system developers. Since we wish to experiment with various novel concepts, we are building our own frame system rather than use one of the commercial systems.

The KFS consists of two layers. Layer 1 provides a very general and flexible framework on top.of which layers with a tighter semantics may be built. It implements frames as an abstract data type and supports object-identity in the standard way. A Frame is an extensible record structure. It has a name and consists of a set of *slots*, each slot has a set of facets. Facets contain data. The semantics of the representation depend on how the higher layers or programs interpret the data in the facets. Data in a facet may be interpreted as values, references to other frames or procedures to mention some possibilities. Layer 2 organizes frames into hierarchies and provides data-types and support for persistence. The hierarchical structuring helps control the complexity of the design process[27]. We will now briefly discuss our design decisions and contrast them with alternative approaches.

2.1. Class and Instance frames

Frames may be of two kinds, *class* and *instance*. As an example, a class frame may define the class **Ships** and its properties, an instance frame may represent the ship **QueenElizabeth**(See Figs. 2, 3).

A given frame is of exactly one kind. Class frames are defined by the user to model the prototypical objects in the world being represented. Slots in a class could represent properties of either the class as a whole or that of its instances. For example the **Ships** class may have the slot **ShipName** which specifies the structure of the instances and the slot **NumberofShips** which is an exclusive property of the class. Any slot in a class frame is required to have the **type**, **value** and **inheritance** facets. It is expected that there will be a moderate number of class frames. We also find it useful to distinguish certain frames as *system frames*. System frames are those class frames which define the system. For example definitions of data types and hierarchy types are system frames. View definitions are another example of system frames. The instance frames represent individuals or the instances of the prototypes. In instance frames, the only facet a slot is required and permitted to have is the **value** facet. It is expected that there will be a large number of such frames. They may be generated as needed from information in the database, as described in the section below on binding and optimization. The *instance* frames need to be implemented carefully from both space and access-time viewpoints.

2.1.1. Inheritance Hierarchies

Frames may reference each other. In addition, the class frames may be organized into *hierarchies*. However a frame may belong to more than one hierarchy at the same time. The **Ship hierarchy** is shown in Fig 2. **NavyShips** and **CommercialShips** are subclasses of the Ships class and may inherit slots from it. Note that each frame in the hierarchy has the slots **ships-parent** and **ships-children**. These are used by the system to store the links to the ancestor and children and are modifiable only through certain functions.

A hierarchy is defined by creating a hierarchy definition frame in the special hierarchy called Hierarchies. A hierarchy definition frame has the following slots: The **parent-slot** stores the name of the parent slot. This slot is required to contain exactly one frame with the exception of the root frame in which it is empty. The **children-slot** gives the name of the children slot. The **root-frame-slot** provides the name of the root frame of the hierarchy. The **hierarchy-slots** are the definitions of all slots which may be inherited through this hierarchy. The **required-instance-slots** are the names of slots which every frame in the hierarchy must possess. Instance frames which are instances of classes in the hierarchy must have values for these slots as a necessary condition for being instances. The **class-slots** give the names of slots which may not be inherited by instance frames. A class-slot cannot be a required-instance-slot. The defining frame for the **Ship hierarchy** is shown in Fig 4. Any frame belonging to

hierarchy possesses the parent-slot, children-slot and all the required-instance-slots for that hierarchy.

Hierarchy Types: Hierarchies may have various semantics. For example, a hierarchy may be a generalization hierarchy. We may further specify properties like disjointness and completeness of subclasses. Hierarchies need not always be generalization hierarchies. For example, a hierarchy could be an aggregation (part-of) hierarchy or a view hierarchy. Such semantics will be enforced by the upper layers in the KBMS.

Inheritance: Any hierarchy-slot may be inherited by a child from its parent in the hierarchy. Inheritance makes it easier to specify similar classes. It often saves the trouble of redefinition, resulting in space saving and ease of maintenance. When we say "slot s is inherited" it means that the slot, all its facets and the data in the facets are also applicable to the child. In general, it may not be appropriate to inherit the slot exactly as it is; we may wish to make changes to the data in some of the facets. This is achieved by updating the desired facets in the child to override the inherited value. The **Inheritance** facet of the slot specifies whether the slot is to be inherited or not. The possible values for this facet are "Yes" and "No".

We now consider the question that at which level the inheritance facet should be specified. There are several choices. The inheritance could be specified **Globally**(with slot definition), at the **parent frame**, or at the **child frame**. We will opt for a **Hybrid** scheme, such that inheritance is specified in the parent frame, which may be overridden by specification at each child. Since a frame may be a child as well as a parent at the same time, an additional **Inheritance-Override** facet is needed to implement the hybrid scheme. In some hierarchies the inheritance may be globally controlled, to simplify automatic interpretation.

A frame may simultaneously belong to multiple hierarchies. However, slots are local to a hierarchy. This defines away the problem of multiple inheritance. The restriction that the hierarchy be a tree structure ensures that there is a single parent of a frame in a hierarchy. An interpreter can switch among the multiple hierarchies, in effect changing interpretive strategy. Such a switch would typically take place when some subgoal has been satisfied: some data have been located, or an intermediate conclusion has been drawn.

What happened to Multiple Inheritance? Multiple inheritance in its full generality is that a frame may have several parents and the value of a slot in a frame is a function (call this function the *inheritance policy*) of the value of the slot in all ancestors of the frame and some other factors (e.g, distance from an ancestor etc.). For example, the value of the completion-date slot in a **projects** hierarchy may be specified to be the minimum of the values in all superclasses.

However, multiple inheritance adds dangerous complexity to the system. The attempt to isolate functions which are useful as inheritance policies is an open ended process and degenerates to allowing the user to write his own inheritance policies as LISP functions, to be executed by a non-sharable interpreter. In general, a dividing line has to be found between what inheritance policies are sharable and defined and and those that are left to an interpreter implementing higher level view definitions or application programs. A problem with arbitrary functions as inheritance policies is that they make optimization by the system difficult.

These problems are better dealt with at a higher level. On the other hand, there are often cases when the same object has different and perhaps apparently contradictory values for a query depending on how we look at it. As an example we may consider an individual's **financial-condition** to be **poor** when he is accessed through the **UniversityEmployees** hierarchy. But it makes sense for his **financial-condition** to be considered **rich** when accessed through the **Students** hierarchy. The problems of management of such information motivated the design decision to consider slots to be local to a hierarchy.

2.2. The Class-Instance Connection

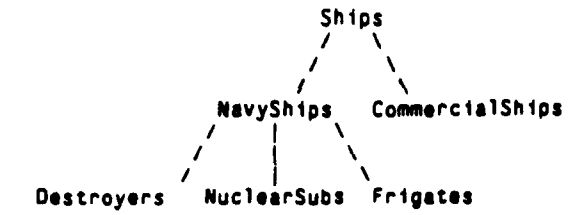
Instance slot: Class and Instance frames are connected by the **Instances** slot in the class frame. Let i be an *instance* frame and c be a *class* frame. A sufficient condition for i to be an *instance-of* c is the occurrence of *i* in the **value** facet of the **instances** slot of c. A necessary condition for i to be an *instance-of* c is that in i, values are assigned (*not* inherited

to all the required-instance-slots of the hierarchy to which c belongs. Another useful facet of the instances slot is the **persistence** facet which is discussed below.

Class-Instance inheritance: The **value** facet is the only facet of slots in instance frames. However, we would like an instance frame to inherit the default value from the class frame if the slot value in the instance is a null value. But we would like inheritance to be suppressed if, for example, the null value has the interpretation *not applicable*, due to the specific structure of the information. It suffices to partition the null values into two sets, the **inheritance-nulls** and **structural-nulls**. As far as this layer is concerned the only relevant semantics of the structural nulls is that they suppress inheritance and therefore it suffices to *not* distinguish them from normal data values. We permit the specification of the inheritance-nulls as a facet called **inheritance-nulls** with each slot. The operational interpretation of this facet is: If the value of a slot in an instance frame is an inheritance-null then replace value with inherited value.

2.3. Persistence

We need to provide a mechanism to store and materialize frames from a persistent secondary store. One could use various storage systems for persistence, like files or relational databases. A **Persistence Hierarchy** provides the information about different available storage systems. The implementation of persistence for instance and class frames will have to be different. We will have a common systemwide mechanism to store and materialize the class frames. However, the structures of instance frames for different classes may vary widely. Therefore, the information about the persistence mapping is best kept on a class by class basis. The **persistence** facet in the instance slot of the class frame refers to a **persistence-frame** which stores this information.




```
( Ships
     ( Ship-parent
          ( Value ))                    ; since no parents
     ( Ship-children
          ( Value NavyShips CommercialShips ))
     ( ShipName
          ( Type Symbol )
          ( Value )
          ( Inheritance Yes ))
     ( NumberOfShips
          ( Type Integer )
          ( Value 1105 )
          ( Inheritance No ))
    ( Instances
          (value  G582)
          (persistence ship-persistence)))

( NavyShips
     ( Ship-parent
          ( Value Ships ))
     ( Ship-children
          ( Value Destroyers Frigates))
     ( #Marines
          ( Type Integer )
          ( Value )
          ( Inheritance Yes )))


( CommercialShips
    ( Ship-parent
          ( Value Ships ))
    ( Ship-children
          ( Value ))
    ( CargoCapacity
          ( Type Integer )
          ( Value )
          ( Inheritance Yes ))
```

Fig 2: The Ship hierarchy and class frames

```
( G582
    (Shipname
        ( Value QueenElizabeth )))
```

Fig 3 : The instance frame for the QueenElizabeth

```
( Ship-hierarchy
     ( parent-slot
               ( Type Symbol )
               ( Value ship-parent )
               ( Inheritance Yes ))
     ( children-slot
               ( Type Symbol )
               ( Value ship-children )
               ( Inheritance Yes ))
     ( root-frame
               ( Type Frame )
               ( Value ships )
               ( Inheritance Yes ))
     ( hierarchy-slots
               ( Type Symbol )
               ( Value ShipName NumberOfShips #Marines
                       CargoCapacity)
               ( Inheritance Yes ))
    ( required-instance-slots
               ( Type Symbol )
               ( Value ShipName )
               ( Inheritance Yes ))
    ( class-slots
               ( Type Symbol )
               ( Value NumberOfShips )
               ( Inheritance Yes )))
```

Fig 4 : The definition of the Ships hierarchy

```
(Ship-persistence

   ( persistence-parent
             (Value Rdbmap))
   ( mapping
             ( Type slotattributemap)
             ( Value
                   (shipname = ships. shipname)))
```

Fig 5 : Example of a Persistence-frame

---

3. Binding and Query Optimization

The persistent data of applications built using the KSYS system are stored in a conventional relational database system. The data must be initially accessed by executing queries to external storage. However, in typical knowledge-based (KB) applications, it does not appear that satisfactory performance can be achieved by simply replacing each subsequent access to the same persistent data by a database retrieval[22]. Such applications can be expected to make many more references to facts during the inferenceing process than are made by the typical database query.

Thus, the brute-force solution of collecting information from secondary storage whenever the application refers to it is too expensive. Our approach is to save accesses to external storage by various techniques; e.g., never repeating the same query (as far as possible). The important idea behind this and other techniques discussed below is **binding**[28] e.g., keeping data in memory within an efficient data structure and establishing links with related data. The various kinds of binding we consider are influenced by our query processing system whose architecture is described below. In particular, the *query optimizer* has many novel characteristics and will be discussed in greater detail.

3.1. Kinds of Binding

To support the goal of accessing the external database as little as possible, we plan to keep information about the past interaction with the database (a query log.) We also need to keep track of various kinds of data already available in main memory. As a special case, the binding of data instances to their parent knowledge frames serves to remind the system that these data are already available in memory, and helps to eliminate redundant accesses to secondary storage.

Some relations may be accessed often enough to justify binding them in their entirety in memory. The relations in which the *metadata* is kept are a good example of this. Generalizing, results of queries are kept in memory as long as it is beneficial. To handle conflicting demands on real memory space (which is finite), it is necessary to use the query log for deciding on the cost/benefit tradeoffs. For transactions which run only for a limited time memory overflow problems should be rare.

An important kind of binding is the *prejoining* of data. Joins along connections as in the *structural model*[29] are good candidates for prejoining, as are joins seen frequently in queries. The resulting join could be kept on external storage, but we plan to only keep them bound in memory. If kept in memory, the join does not need to be materialized by catenation i.e., by copying of the data into new, contiguous, tuples. Instead, we use pointers to the tuples as retrieved into memory from the original relations, along with a descriptor for the result to allow interpretation of the pointers[30]. Such structures are not only space-efficient, but also save on the CPU cost of copying data which has been shown to be significant in memory resident databases[31].

3.2. Query Processor Architecture

The query processor can be distinguished as a module which provides the interface between the frame system and the database system where the persistent data is stored. It consists of three components: Preprocessor, query optimizer, and query interpreter.

Requests for data *queries* posed by the frame system are processed in the following fashion. The preprocessor transforms the query into a relational algebra query (if it is not already in that form). The query optimizer transforms the relational algebra query into an efficient execution plan (this plan is called the Query Evaluation Plan or QEP.) The query interpreter carries out the QEP to produce the results.

The database system is used to perform relation scans and indexed retrievals of data on disk, and some selections and projections. It is *not* used to perform joins; all join processing is done by the query interpreter. The query interpreter manages the memory available to the query processor; it takes hints from the query optimizer e.g., the query optimizer could request that certain relations should be kept memory resident.

Unlike conventional query optimizers, the QEP generated by the query optimizer not only takes advantage of the database structure and the access methods available but also makes use of data bound in memory. This is the reason that the query optimizer must be able to provide hints to the query interpreter.

3.3. Memory Residency

When a large amount of the data needed for query processing is memory resident, we have to take into account a whole new set of costs and goals. In our modeling of costs, it is not sufficient to take only disk accesses into account. It is necessary that CPU costs (comparisons, data movement, evaluation of predicates, hashing) are estimated carefully. As noted above, real memory may not be sufficient to keep all data resident and virtual memory may gain us little versus well-managed secondary storage access. Especially, with increasingly large main memories, memory costs are important and will be taken into account in the cost estimation.

With these considerations, we have developed a comprehensive cost model which can be used by the query optimizer. The cost model spans the range from entirely disk-based query processing strategies to wholly memory resident strategies. The part of the cost model dealing

with memory resident processing is described in[23]. We have identified the important parameters of the cost model, and have developed a cost formula which can be used to account for CPU costs, memory costs, and I/O costs with dynamic weights to reflect changing conditions.

To demonstrate the use of the cost model, we have derived cost equations for various important methods for evaluating a two-way Select-Project-Equijoin on memory resident relations. Analysis of these cost equations leads to some interesting results which are summarized in[32].

3.4. Use of Application Knowledge

Query optimizers must use of a lot of knowledge in their decision making. Some of this knowledge is independent of the particular application e.g, the strategy of performing selections as early as possible is considered to be usually efficient. However, as shown in[17], some of the most effective knowledge used in optimization is application-dependent.

We can classify the knowledge used in query optimization in five broad categories:

- Knowledge which is independent of the database context e.g.,
- $\neg(x > 40) \Rightarrow (x \leq 40)$.
- Knowledge which depends only on the database context, but is independent of the particular database e.g., performing selections as early as possible is usually efficient.
- Knowledge about the application database schema or intension e.g., a relation has a clustered index.
- Knowledge about the actual database contents or extension e.g., cardinality of a relation.
- Knowledge which is dependent on the application e.g., integrity constraints.

The knowledge in the last three categories can be further classified based on the database units it mentions. This classification is given in[23], where we also give examples of each class of knowledge illustrating their use in query optimization. Some of the application knowledge is given in the form of *integrity constraints*.

3.5. Future Work on Optimization

We are currently validating the cost model for query processing with memory resident data. We are doing experiments to derive accurate values for the parameters of the cost model. The query optimizer of the KSYS system will use this cost model. Continuing with our work in utilizing application knowledge, we are currently studying how the query optimizer can uniformly use these kinds of application knowledge along with more domain independent knowledge. We are also setting up an experimental testbed to investigate the utility of various kinds of knowledge.

4. Views

Partitioning of knowledge bases by views is a concept motivated by many of the same considerations which led to the development of views for databases. Knowledge bases will grow to be too large to be conveniently comprehensible to a single person, particularly a casual user. Presenting only a focused subset of the data is a good way to reduce this complexity. Aside from complexity, different groups of users will desire data in a form customized to their own particular needs and interests. Both of these considerations affect the software engineering process. It is much simpler for a program to deal with only the information it needs, without having to sift through irrelevant detail. Also, it is easier for a program to deal

with data in a customized, ususally object-oriented, format, while for purposes of sharing it may be best to store the data in a more general purpose representation. There are several possible approaches to extend the concept of database views to knowledge bases. Since KSYS is an experimental project, we shall probably try several of them.

A special type of hierarchy provides the most direct implementation of views. This type of hierarchy uses somewhat different semantics than those described in Section 2, in order to facilitate sharing across different views. In a universe of persons, there may be hierarchies of taxpayers, family members and employees - representative of the numerous associations in which people engage. KSYS would represent each of these as a separate logical hierarchy, while a given frame will be in as many of these hierarchies as apply. The complications of multiple inheritance are avoided by a restriction that, although frames may participate in many hierarchies, each slot, for purposes of inheritance, may only be in one. Thus an individual would inherit his salary from the employee hierarchy, his marital status from a family hierarchy and so on. Although this is formally less general than a full multiple inheritance scheme, it is our hypothesis that the KSYS scheme will suffice for most applications.

Constraining interpretation by interposing hierarchical views greatly simplifies the design of the interpreters. The construction of an interpreter to operate on general graph structures is very difficult, and must rely either on user guidance[33] or on heuristics[34]. Our interpreters need only be able to process a hierarchy, as opposed to having to navigate through a general graph structure.

Different hierarchies will deal with different sets of frames, organized in different ways, but the frames themselves will always have the same slot values, independent of hierarchy. A given view hierarchy need only focus on a subset of the slots in the frame, but the interpreters have a consistent base to build upon.

Each hierarchy will be a different view of the data, and each user will use whichever hierarchy (or set of hierarchies) correspond to her interests and needs. While multiple hierarchies provide a structure on which to build a convenient user interface, this is not a complete answer to the view problem. Additional theory and structure is needed to facilitate sharing of information across the hierarchies, to ensure consistent updates, and to provide access to the database.

We plan to incorporate the relational model as much as possible in order to take advantage of the existing theory and formalized structure. We incorporate the syntax of relational algebra, letting individual frames represent the various relational operations of **select**, **project**, **join**, and **set difference**, as well as extensions that may be needed for aggregation, temporal inference and recursion.

Subsidiary views may be constructed by means of ordinary relational algebra, at least so far as the data are concerned. However, we also need to be able find an appropriate translation of the non-relational knowledge so that it also is available from the view. A main focus of this part of the project is how to effect such translations. If the additional knowledge is represented as a theory in first order logic, we may take advantage of the duality between relational algebra and logic. The view specification may be written as a set of logic formulas defining new predicates. The theory applying to the view is then that obtained by augmenting the original theory with this new set of formulas. If we wish to manage knowledge such as uncertainty information, which is not easily represented by first order logic, the translation is less direct, and the subject of ongoing research.

In many software engineering and design applications, it is often convenient to use an *object-oriented* paradigm for representing information. However, different classes of users (or the same users at different times) may need this information organized quite differently. Since we hope to share this information, we will have problems with redundancy and update anomalies. It is because of these problems that we feel that objects are not well suited for long term storage of data which must be shared among diverse classes of users[35]. Rather, we wish to use the view facility to create objects as they are needed. Updates to an object must then be translated into updates on the underlying database. This is an example of the view update problem, to which partial solutions exist, either using predefined semantics[16] or ad-hoc techniques suitable for planning-type applications[13].

5. Semantic Integrity Constraints

One important type of knowledge in an integrated knowledge-base system such as KSYS is the collection of semantic integrity constraints controlling persistent data. Two closely related issues arise: the representation of constraints in the semantic hierarchies, and the validity maintenance of constraints in knowledge-base evolution. There can be several semantically equivalent representations of a piece of knowledge which may have quite different computational characteristics in terms of knowledge manipulation. There has been little help from the system on how knowledge should be represented for efficient manipulation. We must be concerned about representing integrity constraints effectively in the hierarchies, taking advantage of the structural information about the organization of the hierarchies and other domain knowledge, such that the validation of constraints will be efficiently performed.

The validity or consistency maintenance of constraints during knowledge-base evolution is vital for the proper functioning of the system. On one hand it is a data engineering issue: the management of the knowledge-base evolution to preserve certain semantics; on the other hand it is also a software engineering issue: the management of the computations (transactions) which manipulate the knowledge-base such that the semantics is preserved. Constraint maintenance has been very difficult to implement efficiently because there is a need for implementation-independent, formal specification of both the constraints and the transactions that may potentially affect the validity of the constraints, while a straightforward evaluation of formally specified constraints against large volume of knowledge and data is very expensive. Furthermore, it is unrealistic to expect that users or programmers understand the semantics of all constraints completely and program them correctly in transactions. We are specifically working on automatic transformations of formal transaction specifications into efficient programs which guarantee the validity of the knowledge-base.

In order to achieve the joint goals of clear specification and efficient excecution, we concentrate on the following two processes[36, 37]:

1. The reformulation of formal constraint specification into representations with better evaluation efficiency;
2. The synthesis of transactions which preserve the integrity constraints of the knowledge-base.

We expand now on these two concepts.

5.1. Constraint Reformulation for Improved Representation

We are developing a framework to exploit knowledge about the application domain and knowledge-base organization to reformulate programmer-specified integrity constraints into representations which are syntactically different but semantically equivalent given the application semantics, and which are more efficient to evaluate in the current knowledge-base configuration. Such a knowledge-based approach provides great potential for more effective constraint representation because: (1) logically equivalent representations can have very different computational characteristics, and (2) after reformulation constraints are specialized to the current computational environment with more optimization opportunities.

Constraint reformulation is accomplished with the technique of antecedent derivation. An inference engine serves as a search space generator of all alternative reformulations. Reformulation is formalized as a tree-search process guided by cost function and control strategies.

5.2. Transaction Synthesis for Efficient Constraint Enforcement

Our approach to the efficient integrity control of large knowledge-bases has several properties: (1) the transactional interface to the knowledge-base is formalized by providing a high-level transaction specification language whose semantics can be formally defined; (2) a formal logic system is developed to reason about the effect of a transaction on the validity of a set of integrity constraints; and (3) transactions are modified to incorporate constraint

validation code efficiently using the property of transaction atomicity and knowledge about transaction control flow and knowledge-base structure.

This approach has many advantages. First of all, both the integrity constraints and the transactions that access the knowledge-base are specified in high-level languages which facilitate knowledge-base programming and software maintenance. Second, synthesized transactions are guaranteed to preserve the validity of the knowledge-base, without going through tedious transaction verification process. Finally, the efficiency of the resulting transaction code is greatly improved by taking advantage of the properties of specific transaction and knowledge-base implementation.

A prototype implementation of constraint reformulation is under development. A generalized theorem prover is used as inference engine. We have implemented the cost model for estimating the evaluation cost of integrity constraints and a set of control strategies for controlling the reformulation process for single constraints. We have just started working on transaction synthesis and have designed algorithms for constraint regression through primitive update operations.

6. The Semantic Mediation of Operations across Disparate Domains

It is often necessary for a decision-making support system to obtain data from a variety of sources so that decisions can be based on a a fusion of a broad range of information. These sources may include relations and relation fragments within a single distributed database system to several homogeneous or heterogeneous databases or multidatabase systems. Particularly where information must be obtained from several database systems owned by distinct user groups, it may not always be represented in forms that are directly comparable.

Within all of these contexts, as well as within a single database system, it remains desirable to be able to perform operations across domains that are seemingly incompatible, say, because of type conflicts, but that are in some fundamental sense semantically similar or the same. While this requirement is most clearly pronounced in the multidatabase situation[38], the need sometimes also arises within a single database system, as, for example, when its conceptual design is not adequate to the demands placed upon it for generating information according to new views being imposed on the old data.

At the same time, such situations also point to the need for some sort of constraint system to guard against semantically meaningless operations in cases where domains may superficially appear to be the same, but in fact are not. If a user who is unaware of the semantic intention of the implementation performs operations across disparate domains, the results may not be meaningful. Domain mismatch occurs when two attributes are not of the same types, or when they are of the same types, but the types have a different semantics in some of the relations involved in the operation.

This aspect of KSYS investigates methods for the correct performance of relational database query operations despite data domain mismatch. It addresses problems related to distributed database systems and applications where information must be extracted from heterogeneous domains. One means of ameliorating this problem is through the use of an abstract type facility; this can provide a semantic classification of domains, independent of their representation. An abstract type facility can thus be used to exclude meaningless operations between semantically incompatible domains when these are represented by different types. The extent to which a type system can perform this function depends on the notion of type equivalence that it uses; we assume that name equivalence is used, so that two data types are considered equivalent only if they are from the same database system and bear the same name.

The question naturally arises as to how one may allow for operations between types that are not equivalent but that do have a common core of semantic meaning. The traditional varieties of type conversion in programming languages are insufficient for this task. Our model for the semantic mediation of operations across *disparate domains* is based on the notion of *virtual attributes*. A virtual attribute denotes the property of some entity and is associated with a particular domain. The virtual attribute need not be physically present in the database although it might be materialized for efficient processing. Virtual attributes are derived from

real attributes by invoking functions or mappings which embody knowledge of the semantic relationships between domains.

Our work is proceeding along the following directions:

- A formalization of operations across disparate domains in terms of operations in the presence of incomplete information. We propose an extension of the relational operators to handle operations across virtual attributes. Theoretical and practical limitations of these extended operations are being considered.
- A consideration of the impact of virtual attributes and domain mediation on query-processing efficiency, including means for avoiding loss of performance.
- In view of the theoretical limitations on the performance of operations across disparate domains, we are also examining heuristic approaches for use when more formal methods fail.
- Future work will also address the implementation of a frame-based system residing at a level above the database schema to provide a virtual attribute facility. This system is intended to embody both a functional encoding of knowledge of domain relationships in terms of a virtual attribute facility and a heuristic component to handle problems of incompleteness.

7. Kaleidoscope: Open-Ended Menu Interface for Data Access

The large data collections that the KSYS approach envisages are difficult for a user to understand. The use of a formal language, such as SQL, provides a formal and clean interface, but does nothing to mitigate the underlying complexity. We wish to exploit knowledge about the data to help the users. A user's difficulties in accessing a database often causes additional programming effort, requiring an intermediary programmer, who hand-crafts a solution to deal with the user's inability to extract the desired information. The Kaleidoscope project is concerned with providing an open-ended, knowledge-driven menu interface that ultimately reduces the work load for software engineers.

Using an artificial query language for problem solving, whether it is formal like SQL or whether it mimics natural English, involves scheduling of two cognitive processes: (1) the *planning process* that formulates the mental propositional content of a query and produces it in a query language, and (2) the *delivery process* that ships the outcome of the planning process to the system. Human short term memory is not big enough to hold a query of typical length and becomes a major bottleneck to these processes. In addition, the unreliable process of human long-term memory retrieval[39] interferes with users in planning and delivery of queries. Naive users are further subject to the difficulties[40] associated with learning a new language and the logical data organization of a system. Some knowledge-based systems have incorporated menu-driven dialog interfaces, but these interfaces are typically geared to a particular task and their capability is prespecified by an articulated task model. The open-ended nature of problems to be solved demands maintaining interfaces that allow arbitrary access and manipulation of data.

Kaleidoscope is a query interface using knowledge-driven, context-dependent menu system. A linear syntax query language is the medium for casting a system's underlying semantic power over menus. The system predictively generates a collection of choices on a menu, and the user constructs a query incrementally via a sequence of guided selections. Kaleidoscope provides two alternatives for the user query language: (1) SQL for direct manipulation of database objects and; (2) a subset of natural language for access of the interpreted data at high level. Kaleidoscope's use of menus is based on previous work on NLMenu[41] and INGLISH[42]. Menus guide the users within an acceptable boundary of the system's limited syntax and semantics.

7.1. Linear Menu-Guided Query Creation

The simplest way of creating a query is to follow the menu guidance linearly. To illustrate, shown below is a SQL query created in 12 successive mouse clicks on selected choices. Numbers indicate the sequence of selections.

SELECT$_1$ Ship-Name$_2$, Owner$_3$, Weight$_4$ FROM$_5$ Ships$_6$ WHERE$_7$ Weight$_8$ $\geq_9$ 100000$_{10}$ ORDER BY$_{11}$ Weight$_{12}$

Each choice comes from a different state of a dynamically changing menu window. SQL delimiters like commas and parentheses are automatically inserted by the system, whenever it is obvious by later selections. A small status window is allocated on the screen to display a partial query being constructed and a cursor indicating where next action will take place. To create terminal symbols such as database values, number constants, and database object identifiers (like tuple variables and view names) that should be created by the users, Kaleidoscope follows NLMenu's approach by generating a set of demons on the menu, each of which, once selected, prompts a nested interaction with the user. These demons form a special class of preterminal symbols.

7.2. The Interpreter

To generate choices predictively on the menu, the Kaleidoscope interpreter operates over a domain-independent grammar, which is coupled tightly to the domain-specific semantics stored in a knowledge base. At run time, the interpreter maintains a *dynamically changing forest* of parse trees with shared terminal and nonterminal nodes. Most of the trees in the forest are incomplete during query construction. As the user makes a choice, the interpreter aborts the trees that cannot incorporate the new choice as part of their body and then expands the forest to find the next set of choices. If there are new complete trees in the forest, the interpreter enables query execution. For an unambiguous query language, the number of such complete parse trees is at most one. It is this dynamically changing forest of trees that distinguishes Kaleidoscope from the conventional menu system which stores its choices in a predetermined hierarchy, and gives the expressive power of a context-free language to Kaleidoscope.

7.3. More Flexibility in Menu Interaction

So far, we introduced a style of menu interaction that is linear with respect to the terminal symbols of acceptable queries. However, this style of interaction burdens the users with navigation in a limited choice space, since mental query planning does not necessarily match the left-to-right linearity of a query. We see two ways of handling this problem: (1) accommodating a limited range of syntactic variety for natural language queries; (2) providing more flexibility in menu interaction. A full range of syntactic variety is avoided in favor of keeping the grammar for natural language queries small. Two specific functions for flexible menu interaction are presented next.

Kaleidoscope allows a choice selection to be deferred until the later part of a query is constructed. The system recognizes how far to go in deferment by the nonterminal path to current choices in the forest. When the user returns to the deferred point by a system command, the interpreter takes into account the additional selections made in between. For example, it is often convenient in SQL to defer the choice of projection attributes in a SELECT clause until the relations are specified in a FROM clause or later. When the user returns to the SELECT clause after completion of the FROM clause, the system considers only the attributes of the selected relations to be generated. The user can also move the cursor back to an arbitrary point of the current query to redo previous selections. Consistency checking is triggered for the rest of the modified query. This is also a way to create quickly a query whose structure is similar to one of the previous queries. The system maintains a history of recent queries and allows the user to retrieve a query into the status window.

7.4. Activation of Stored Knowledge

Activation of semantic constraints in predictive choice generation is essential for effective menu-guidance. Syntactic knowledge of a language, for instance a BNF definition of SQL, does not provide enough constraining power. The result is a flood of irrelevant choices on menus and creation of semantically anomalous queries. To make the knowledge base accessible to the interpreter, arbitrary constraint predicates are attached to context-free grammar rules. Demons which can access database values use stored information to guide users in creating database values in query, by presenting a list of values for the user to choose or asking the user to enter a value based on a suggestion on its range.

7.5. Benefits ar Status

Currently, we are implementing the interpreter of Kaleidoscope. Written in Common Lisp, the interpreter will be integrated to operate on top of the KSYS frame system.

We conjecture that Kaleidoscope will replace human lower level query planning and delivery process by menu choice search and selection. This means the human users are guarded from the trivial errors[40] in planning and delivery. In addition, once structural misconceptions are filtered the post-query cooperative response can focus on the other types of misconceptions due to erroneous assumptions of content[15, 12]. In Kaleidoscope the semantics of the underlying system are pushed upward to the process of generating surface query fragments on the menu. This contrasts with the typical approach of formal and natural language query interfaces, in which the flow of control follows a top-down translation and the semantic power of a system is not visible to users. Kaleidoscope needs only a small number of syntactic rules and vocabularies. This distinguishes Kaleidoscope from the approach of relatively passive natural language interfaces like TEAM[43], that require a large number of syntactic rules and vocabularies.

8. Knowledge Acquisition from Databases

The development of knowledge bases is a serious data engineering problem, hindered particularly by the bottleneck of knowledge acquisition and validation. Building knowledge bases is expensive, time consuming, and often yields unsatisfactory results. In addition, knowledge base maintenance is difficult because modifications to a knowledge base may cause cases that previously were handled correctly to fail. We need new data engineering tools to help break the knowledge acquisition bottleneck, and to simplify knowledge base management. We are currently developing a program to assist in knowledge acquisition and management by using databases of case examples.

The knowledge acquisition subsystem is intended to assist in expert system applications in which domain experts give estimates of the probabilities of events, and the probabilistic relationships between events. There are many applications where probabilities are necessary or desirable. The notion of uncertainty may apply to either the likelihood of occurrence of an event or the strength of a relationship as indicated by the measures used to record an event. Knowledge engineers have found difficulty in implementing systems using uncertainty, because experts have great difficulty providing reliable estimates of probabilities. Estimates are often inconsistent among experts, and individual experts are often not self-consistent, and often experts cannot provide any quantifiable estimate at all, particularly for joint probabilities of multiple events.

Integrating databases which record cases of events with knowledge acquisition methods can help solve these problems. After a domain expert has entered initial knowledge, if databases containing case examples are available, a computer program can examine the database to validate the knowledge entered by the expert, and, where appropriate, propose additional knowledge to incorporate. Databases usually contain more cases than any one expert has seen. Programs can analyze a case database to determine probabilities more accurately and completely than the expert.

The 1986 Banff Workshop on Knowledge Acquisition for Knowledge-Based Systems[44]

provides a recent overview of knowledge acquisition research. Research in automated discovery has been particularly concerned with using empirical databases for knowledge acquisition; we reviewed these systems in[45]. The RX and RADIX projects,[46, 47] were earlier projects from our group are predecessors of our current work. In the RX and RADIX projects we developed a program to discover and confirm medical knowledge from databases by a combination of statistical and knowledge-based methods. RX performed as follows. It generated a hypothesis relevant to the database (using simple correlation), e.g. "the drug prednisone increases serum cholesterol". It evaluated the hypothesis by extracting appropriate data from the case database, and designing and executing a statistical analysis. It interpreted the results to determine validity, and incorporated validated knowledge in the knowledge base.

RX was a relatively successful experiment. It re-discovered medical relationships not previously known to the project participants. It provided new, independent evidence of suspected relationships. It accurately confirmed known medical relationships. Most important, it demonstrated that a combination of statistical and knowledge-based methods are effective in finding and confirming relationships from a case database.

We plan to apply and extend these capabilities to the specific problem of knowledge validation and acquisition. The current design is to perform as follows.

1. The domain expert and knowledge engineer create a preliminary knowledge base which includes probability estimates for relationships.
2. The system either automatically or jointly with the user selects relationships from the knowledge base to confirm against the database of case experience. Relationships may be selected for confirmation because of their high frequency of use in rules, because of low confidence indicated by the user, or for their importance in reaching conclusions (e.g. as indicated by sensitivity analysis). There are usually probabilities that the developer does not know; he may tell the system to determine these.
3. The program designs an appropriate statistical study to confirm or find the probabilities. This includes determining if there is sufficient data in the database to perform the analysis, determining if confounding variables are known, determining methods to control for confounding variables, and selecting a suitable statistical method (e.g,. cross sectional analysis, longitudinal regression, etc.).
4. The program carries out the design specified in step three, by extracting the relevant cases from the database and performing the statistical analysis.
5. Finally, the program reports those relationships that were found to be accurately described, those that differed from the expert, and those that the expert did not specify which it was able to determine.

In addition to helping in initial knowledge acquisition, this capability is useful for validating knowledge bases that have been modified. When new knowledge is added, it is difficult to confirm that the system still runs correctly for cases that previously worked. It would be useful to provide a simple method to validate knowledge base modifications; to test them by some automated means. This is a capability we plan to include; to use the case database as a reservoir of test cases to validate additions or modifications to the knowledge base.

Performing such knowledge validation and acquisition within KSYS should strengthen the confidence that users must have if KBMSes are to be widely accepted.

9. Postscript

The engineering of AI systems remains a major problem. Our KSYS project is intended to elucidate and demonstrate methods to deal with the encoding and interpretation of conceptual knowledge and factual data. It is important to develop techniques that will engender trust in systems using knowledge-based techniques. Shared use of knowledge is important for economy but also as a means of sharing concepts, and assuring consistency of interpretation of the large

database on which modern planning systems are based.

To have any hope of success the structures must be composed of simple and understandable components, as we have learned from our experience in dealing with large quantities of factual data. The knowledge problem is harder. The methods still must not demand such a level of sophistication of the users that the systems cannot be understood or properly managed. We hope not to give up too much power to achieve this goal.

The focal concept guiding our research is the imposition of views on knowledge- and databases. These views define application areas, object structures, and the scope for automated inference procedures. Integration of these views will lead to a complexity which will be hard to manage, so that we plan to always restrict maintenance and control effort to one active view at a time. We hope to able to demonstrate within KSYS that such a partitioning will permit retention of most of the power of fully general knowledge networks, while introducing reliability, maintainability, and adaptability.

10. Acknowledgements

Basic research in the Management of Knowledge and Databases is supported by ARPA under contract N39-84-C-211, managed by SPAWAR. Research on theory and computation for natural language processing was supported by NSF-IST grant 8023889 and was in part aided by Texas Instruments. Related work on RADIX was supported by the National Library of Medicine through grant LM-04334. Earlier funding was provided by the National Center for Health Services Research through grant HS-04389. Computing resources are partially supported by Intellicorp, DEC ISTG and by SUMEX-AIM, which is funded by NIH grant RR-00785 from the Biotechnology Resources Program. As always, our conclusions do not reflect official positions of any of our sponsoring agencies.

References

1. Barr, A., Feigenbaum, E. A., *The Handbook of Artificial Intelligence*, HeurisTech Press. Stanford, California, , Vol. 1, 1981.

2. Codd, E.F., "A Relational Model for Large Shared Data Banks," *CACM*, Vol. 13, No. 6, 1970, pp. 377-387.

3. Kim, W., Reiner, D., and Batory, D., *Query Processing in Database Systems*, Springer-Verlag, New York, 1985.

4. Brodie, Mylopoulos and Schmidt, *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies*, Springer-Verlag, 1987, Proceedings of Islamorada Workshop, Feb. 1985, Computer Corporation of America.

5. Kerschberg, L., *First Workshop on Expert Database Systems (EDBSW)*, , 1986, To be published as Expert Database Systems by Benjamin Cummins.

6. Schkolnick, M. and Tiberio, M, "Estimating the Cost of Updates in a Relational Database," *ACM TODS*, Vol. 10, No. 2, 1985, pp. 163-179.

7. S. Weyl, J. Fries, G. Wiederhold, and F. Germano, "A Modular Self-Describing Databank System," *Computers and Biomedical Research*, Vol. 8, 1975, pp. 279-293.

8. Gadia, S.K., "Towards a Multi-Homogeneous Model for a Temporal Database," *Proc. IEEE Data Engineering Conference*, Feb 1986.

9. Bancilhon,F.B. and Spryatos,N., "Update Semantics of Relational Views," *ACM TODS*, Vol. 6, No. 4, 1981, pp. 557-575.

10. Suwa,M., Scott,A.C., and Shortliffe,E.H., "An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System," *The AI Magazine*, Vol. 1, No. Fall, 1982, pp. 16-21.

11. Wiederhold, G.C.M., Blum, R.L. and Walker, M. G., "An Integration of Knowledge and Data Representation," in *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies*, Brodie, Mylopoulos and Schmidt, eds., Springer-Verlag, 1987, Proceedings of Islamorada Workshop, Feb. 1985, Computer Corporation of America.

12. F. Corella, S. J. Kaplan, G. Wiederhold, and L. Yesil, "Cooperative Responses to Boolean Queries," *Proc. IEEE Data Engineering Conference*, April 1984, pp. 77-93.

13. J. Davidson and S. J. Kaplan, "Natural Language Access to Databases: Interpreting Update Requests," *American Journal of Computational Linguistics*, Vol. , No. , April-June 1983, pp. 57-68.

14. S. Finkelstein, "Common Expression Analysis in Database Applications," *International Conference on Management of Data*, ACM-SIGMOD, June 2-4 1982, pp. 235-245.

15. S. J. Kaplan, "Cooperative Responses from a Portable Natural Language Query System," *Artificial Intelligence*, 1982, pp. 165-187.

16. Arthur M. Keller, "The Role Of Semantics in Translating View Updates," *Computer*, Vol. 19, No. 1, January 1986, pp. 63-73.

17. J. King, "QUIST: A System for Semantic Query Optimization in Relational Databases," *Proc. of the 7th Conference on Very Large Data Bases*, Zaniolo and Delobel, eds., 1981, pp. 510-517.

18. N. Rowe, "An Expert System for Statistical Estimates on Databases," *National Conference on Artificial Intelligence*, March 1983, pp. .

19. Wiederhold, G.C.M., "Knowledge and Database Management," *IEEE Software*, Vol. 1, No. 1, 1984, pp. 63-73.

20. McDermott, J., "R1: A Rule-Based Configurer of Computer Systems," *Artificial Intelligence*, Vol. 19, No. 1, 1982.

21. Stonebraker, M., "The Design of the POSTGRES storage System," *Proceedings 13th VLDB Conference*, 1987, pp. To appear.

22. Ceri, S., Gottlob, G., and Wiederhold, G., "Interfacing Relational Databases and Prolog Efficiently," *Proceedings of the First International Conference on Expert Database Systems*, 1986, pp. 141-153.

23. Swami, A., "Application Knowledge and its Use in Query Optimization," Tech. report, Stanford University, 1987.

24. Duda, R.O., and Shortliffe, E.H., "Expert Systems Research," *Science*, Vol. 220, 1983, pp. 261-276.

25. M. Minsky, "A Framework for Representing Knowledge," in *Mind Design*, Hangeland, J., ed., The MIT Press, 1981, pp. 95-128.

26. R. Fikes and T. Kehler, "The Role of Frame-Based Representation in Reasoning," *CACM*, Vol. 28, No. 9, September 1985, pp. 904-920.

27. E.W. Dijkstra, "The Structure of the THE MultiProgramming System," *CACM*, Vol. 11, No. 5, 1968, pp. 341-346.

28. Wiederhold,G.C.M., "Binding in Information Processing," Tech. report, Stanford University, 1981, Report STAN-CS-81-851

29. Wiederhold, G.C.M., and El-Masri, R., "The Structural Model for Database Design," *Proceedings of International Conference on Entity-Relationship Approach*, 1979, pp. 247-267.

30. Lehman, T.J., and Carey, M.J., "Query Processing in Main Memeory Database Management Systems," *Proceedings of ACM-SIGMOD International Conference on Management of Data*, 1986, pp. 239-259.

31. Bitton, D., Hanrahan, M.B., and Turbyfill, C., "Performance of Complex Queries in Main Memeory Database Systems," *Proceedings of IEEE Data Engineering Conference*, 1987, pp. 72-81.

32. Swami, A., "A Cost Model for Memory Resident Databases," Tech. report, Stanford University, 1987, Submitted for publication.

33. Stonebraker,M. and Kalash,J., "TIMBER: A Sophisticated Relation/Browser," *Proceedings 8th VLDB Conference*, McLeod and Villasenor, eds., 1982, pp. 1-10.

34. Korth,H.F., Kuper,G.M.,Feigenbaum,J., vanGelder,A., and Ullman,J.D., "System/U: A

Database System Based on the Universal Relation Assumption," *ACM TODS*, Vol. 9, No. 3, 1984, pp. 331-347.

35. Wiederhold, G.C.M., "Views, Objects, and Databases," *IEEE Computer*, Vol. 19, No. 12, December 1986, pp. 37-44.

36. Qian, X., Smith, D.R., "Constraint Reformulation for Efficient Validation," *Proc. 13th Int'l Conf. VLDB*, , September 1987, pp. to appear.

37. Smith, D.R., "Derived Preconditions and Their Use in Program Synthesis," in *Sixth Conference on Automated Deduction*, D.W.Loveland, ed., Springer-Verlag, New York, 1982, pp. 172-193, Lecture Notes in Computer Science 138.

38. Litwin,W. and Vigier, P., "Dynamic Attributes in the Multidatabase System MRDSM," *Second Int. Conf. on Data Engineering, IEEE CS*, 1986.

39. Stuart K. Card, Thomas P. Moran, and Allen Newell, *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates, 1983.

40. Phyllis Reisner, "Human Factors Studies of Database Query Languages: A Survey and Assessment," *ACM Computing Surveys*, Vol. 13, No. 1, 1981, pp. 13-31.

41. Craig W. Thompson, Kenneth M. Roth, Harry R. Tennant and Richard M. Saenz, "Building Usable Menu-Based Natural language Interface to Databases," *9th Conf. on VLDB*, 1983, pp. 43-55.

42. Brian Phillips and Sheldon Nicholl, "INGLISH: A Natural Language Interface," *Foundation for Human-Computer Communication*, K. Hopper and I. A. Newman, eds. IFIP WG 2.6 Working Conference on The Future of Command Languages, 1985, pp 7-26.

43. Barbara Grosz, Douglas E. Appelt, Paul Martin, and Fernando Pereira, "TEAM: An Experiment in the Design of Transportable Natural Lnguage Interfaces," Tech. report 356, SRI AI center, Aug 1985.

44. *Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Alberta, Canada, 1986, Selected articles to appear in the International Journal of Man-Machine Studies.

45. Walker, M.G., "How Feasible is Automated Discovery?," *IEEE Expert*, Vol. 2, No. 1, Spring 1987, pp. 70-82.

46. Blum, R. L., "Discovery, Confirmation, and Incorporation of Causal Relationships from a Large Time-Oriented Database: The RX Project," *Computers and Biomedical Research*, Vol. 15, No. 2, 1982, pp. 164-187.

47. Walker, M.G., and Blum R.L., "Towards Automated Discovery from Clinical Databases: the RADIX Project," *Proceedings of the Fifth World Congress on Medical Informatics (Medinfo)*, Elsevier Science Publishers, 1986, pp. 32-36.

APPLICATIONS OF OBJECT-ORIENTED DATABASE TECHNOLOGY IN KNOWLEDGE-BASED INTEGRATED INFORMATION SYSTEMS

Frank Manola

Computer Corporation of America

Contributed by Frank Manola

Applications of Object-Oriented Database Technology in Knowledge-Based Integrated Information Systems*

Frank Manola
Computer Corporation of America
Cambridge, Massachusetts

Abstract

Knowledge-based integrated information systems denote a class of systems that involve the integration of heterogeneous information sources. These information sources may include heterogeneous distributed database systems, knowledge-based systems (such as expert systems) involving heterogeneous knowledge representations, and conventional application programs and their associated processors. This paper briefly describes several applications of object-oriented database technology in knowledge-based integrated information systems, including the integration of heterogeneous (and distributed) system components, and heterogeneous data and knowledge representations, and identifies some future research directions for the underlying technology.

Introduction

Generality and flexibility are increasingly important system design criteria, for practical, not theoretical, reasons. The need for generality is increasingly motivating the integration of pre-existing systems within organizations, both to enable better use to be made of existing data, and to enable management to assert greater control over valuable data resources. The need for flexibility arises because organizations must be adaptive to changing business situations, hence their systems must be adaptive. In addition, flexible systems allow for reduction of system life-cycle costs in the face of inevitable changes in technology.

Database Management Systems (DBMSs) were originally introduced to provide generality and flexibility in the context of business data processing. Prior to the introduction of DBMSs, data was dispersed among multiple applications, organized according to the specific needs of individual applications, and was inaccessible except through those applications. This meant that it was difficult to gain access to data, or to organize it, for new and unanticipated purposes. There was also considerable redundancy, both of data, and of program code required to manipulate and maintain the multiple application files. The database substituted a central data repository, and accompanying data management facilities, equally accessible by multiple applications. This reduced data redundancy, as well as redundant data manipulation program code previously required within the applications. *Object-oriented database technology* is being developed to extend this generality and flexibility to "unconventional" types of data, and associated processing, including text, graphics, and voice data, that cannot currently be handled by DBMSs, and that thus cannot be integrated in the same way as conventional data.

Object-Oriented Concepts

Object-oriented concepts were originally introduced in the programming language and artificial intelligence communities. While there are many variations of the object-oriented approach, a typical description (using typical terminology [COX86]), is as follows. The phrase *object-oriented* generally refers to the idea of decomposing a system into a collection of logical or physical entities called

* This work was partially supported by the Defense Advanced Research Projects Agency and by the Space and Naval Warfare Systems Command under Contract No. N00039-85-C-0263. The views and conclusions contained in this paper are those of the author and do not necessarily represent the official policies of the Defense Advanced Research Projects Agency, the Space and Naval Warfare Systems Command, or the U.S. Government.

objects. Conceptually, an object is a self-contained collection of data and procedures that can access that data. The data is private to the object, and cannot be accessed except via one of that object's procedures. The procedures support a set of operations (the object's *protocol*) that are visible outside the object. Users invoke an operation by sending a *message* to the object effectively asking it to perform the operation. On receiving a message, the object selects and performs one of the procedures, called a *method*, corresponding to the message. This involves a dynamic binding of the name of the operation (the *selector*) with the code to be executed. Objects with the same methods are grouped into *object classes*, and the methods are often physically grouped with the object class definition in order to reduce redundancy. However, the data associated with a particular object (called the *instance variables*) remains with the object. An object-oriented system may be extended by defining new object classes, and these new object classes may be defined in terms of existing object classes. A particular form of definition of a new object class in terms of existing ones is when a new object class is defined as a *specialization* of an existing one. In this case, the object classes are said to form an *inheritance hierarchy*, and the specialized class is said to *inherit* the methods of the more general class (i.e., the methods of the general class also apply to the specialized class). Methods may also be defined for the specialized class that apply only to that class, or that extend or replace the methods it inherits.

Figure 1 illustrates some simple objects in an object-oriented model. The top left object represents a **Part** object class that might be defined for a CAD application. The object class has **display**, **weight**, **specification**, and **part_number** methods. The top right object represents a **Part** object (instance), with its associated method calls and instance variables. The bottom left object represents the object class for a specialized type of **Part** object, a **3Dsolid** object. The **3Dsolid** object

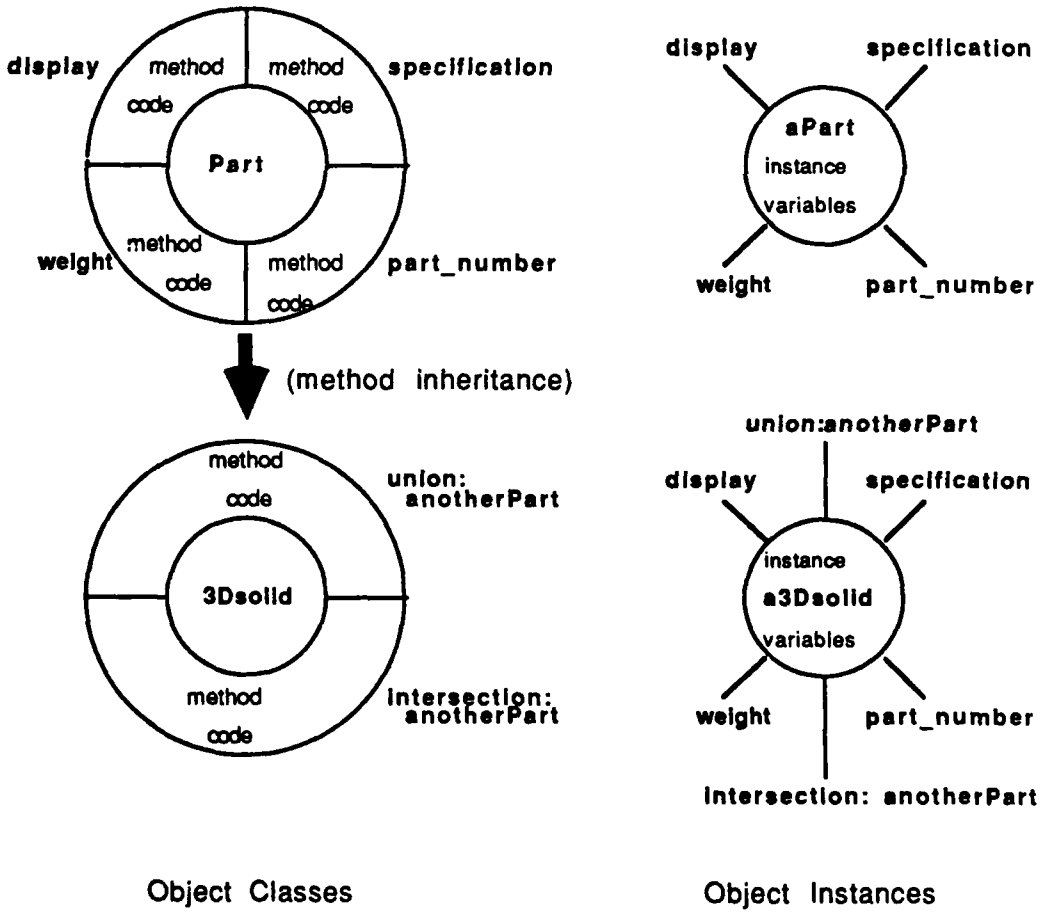


Figure 1 Object Classes and Objects in an Object-Oriented Model

class defines two additional methods that apply only to **3Dsolid** objects, **union** and **intersection** operations. When directed at a given **3Dsolid**, these operations return the **3Dsolid** that represents the union or intersection, respectively, of the given object and the **3Dsolid** object supplied as a value of the **anotherPart** parameter. Only the two new methods are defined for object class **3Dsolid**. However, because it is a specialization of object class **Part**, the **3Dsolid** object (instance) on the bottom right is shown as inheriting all the method calls that apply to **Part** objects, together with the specialized ones defined for **3Dsolid** objects.

Object-oriented DBMSs [LOCK85, DITT86] generally attempt to capture these same concepts, but also emphasize certain additional characteristics necessary to support large, shared, persistent object stores. These characteristics include efficient processing of set-oriented requests (query optimization), efficient processing over large secondary storage organizations, concurrency control, and recovery.

Compared to a conventional relational DBMS, a typical object-oriented DBMS differs primarily in trying to support user-defined operations on object classes, and (method) inheritance. Other aspects found in object-oriented DBMS descriptions include support for the concept of *object identity* (the object has an identity independent of the values of its instance variables), and *direct object relationships* (objects related to a given object are accessed by invoking a method of the given object, as if the related objects were part of the given object's internal data). Representative examples of current OODBMS work include EXODUS [CARE86], GEMSTONE [MAIE86], PROBE [DAYA86, MANO86a,MANO86b,ROSE86], CACTIS [HUDS86], GARDEN Object Server [SKAR86], and POSTGRES [STON86b]. With the exception of GEMSTONE, none of these systems is an announced product.

An Example Object-Oriented Database System

We briefly describe the characteristics of CCA's PROBE system, currently under development, in order to illustrate the capabilities of one specific object-oriented database system. PROBE was primarily intended to support applications involving such "nontraditional" data types as spatial, image, and time data. PROBE incorporates an object-oriented data model that allows integration of programs implementing object behavior together with data, is extensible, and allows the use of special-purpose processors for new data types. It provides efficient recursive query processing (for use in queries on parts hierarchies or task precedence networks), as well as spatial and temporal query processing (using special object types).

PDM, PROBE's data model, is an extension of the Daplex (functional) model [SHIP81]. Daplex already contained the concept of "entities" (objects) to which "functions" (messages) could be applied in a dynamic way, inheritance hierarchies of entity types, and other characteristics similar to those of an object-oriented model. Daplex was also the basis of several implemented CCA systems, including the ADAPLEX LDM/DDM, a DBMS implemented in, and designed to work with, the Ada* programming language, and the MULTIBASE heterogeneous distributed database system, designed to integrate multiple existing database systems into a single system, so there was implementation experience available with at least some aspects of an object-oriented DBMS in developing PROBE. The basic areas in which Daplex has been extended in PDM are:

- multiargment and computed functions are supported
- a collection of operations, called *PDM algebra*, is defined as the basis for defining system operations
- the ability to define new object classes with specialized operations (extensibility) is supported

PDM contains two basic concepts: *entities* and *functions*. An entity is a construct that denotes some individual thing (and thus corresponds to an object, as described earlier). Entities with

---

* Ada is a trademark of the Department of Defense (Ada Joint Program Office)

similar characteristics are grouped into collections called *entity types*(e.g., **PART**). Properties of entities, relationships between entities, and operations on entities (behavioral aspects) are all uniformly represented in PDM by *functions* (which correspond to methods). In order to access properties of an entity or other entities related to an entity, or to perform operations on an entity, one must evaluate a function having the entity as an argument. The examples below illustrate some of the aspects of PDM functions (the syntax used is illustrative only):

• the single-argument function **PART_NUMBER(PART) --> STRING** allows access to the value of the part number attribute of a **PART** entity.

• the multi-argument function **COLOR(X,Y,PHOTO) --> COLOR_VALUE** allows access to the color values at particular points in a photograph.

• the function **LOCATION(CONNECTION,LAYOUT) --> (X,Y)** allows access to the value of the coordinates of a connection in a diagram (note that a function can return a complex result).

• the function **UNION(3DSOLID,3DSOLID) --> 3DSOLID** provides access to a union operation defined for 3-dimensional solid models of parts.

• the set-valued function **COMPONENTS(PART) --> set of PART** allows access to the component parts of a group part (assembly).

Functions may also be defined that have no input arguments, or that have only Boolean (truth-valued) results. For example:

• the zero-argument function **PART() --> set of ENTITY** is implicitly defined for entity type **PART**, and returns all entities of that type (a corresponding function is implicitly defined for each entity type in the database).

• the function **ADJACENT(PART,PART,ASSEMBLY) --> BOOLEAN** defines a predicate that is true if two parts are adjacent within a given assembly. All predicates within PDM are defined as Boolean-valued functions.

Functions in PDM may be either *intensionally-defined*, with output values computed by procedures (such as the **UNION** function above), or *extensionally-defined*, with output values determined by a conventional database search of a stored table (such as the **PART_NUMBER** function above).

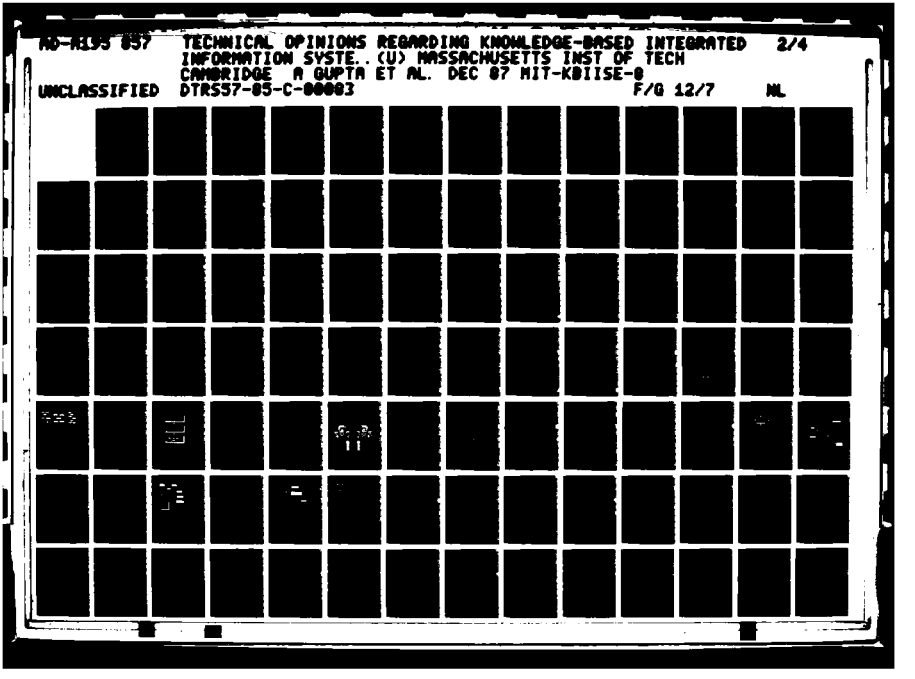
Specializations (subtypes) of entity types may be defined, forming an *inheritance (isa) hierarchy*. For example, the declarations:

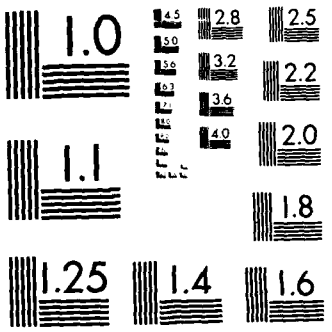
entity PART
function PART_NUMBER(PART) --> STRING
function WEIGHT(PART) --> REAL

entity 3DSOLID isa PART
function UNION(3DSOLID,3DSOLID) --> 3DSOLID

define entity types similar to the object classes shown in Figure 1. They define a **PART** entity type having two functions, and a subtype **3DSOLID** having an additional function. Because **3DSOLID** is a subtype of **PART**, any **3DSOLID** entity is also an entity of the **PART** supertype, and automatically "inherits" the **PART_NUMBER** and **WEIGHT** functions.

Generic operations on objects (entities and functions), such as selection, function application, set operations, and formation of new derived function extents, have been defined in the form of an algebra [MANO86b] similar in many respects to the algebra defined for the relational data model. Like the relational algebra, this *PDM algebra* provides a formal basis for the definition of general database operations. In particular, the algebra serves to define the semantics of expressions in


AD-A195 857 TECHNICAL OPINIONS REGARDING KNOWLEDGE-BASED INTEGRATED 2/4
INFORMATION SYSTE.. (U) MASSACHUSETTS INST OF TECH
CAMBRIDGE A GUPTA ET AL. DEC 87 MIT-KBIISE-8
UNCLASSIFIED DTRS57-85-C-00083 F/G 12/7 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

PROBE's query language, *PDM Daplex,* involving functions and entities, such as:

```
for P in PART, for D in DRAWING
  print(PART_NUMBER(P)) where
    WEIGHT((P) > 5 and
     SQ_INCHES(AREA(P,D)) < 10
```

Integration of Heterogeneous Components Using the Object-Oriented Approach

Object-oriented approaches provide a very natural framework for use in integrating heterogeneous components. As a *design approach*, thinking of the components to be integrated as objects allows a common methodology to be applied to objects at all levels of granularity. Whether the objects to be integrated are entire systems, or individual object classes within a single system, the approach has a number of important effects:

- It focuses attention on the definition of the message protocols that give access to the object behavioral semantics, as implemented by the object methods. In a distributed environment, development of such protocols and support for them would have to be done anyway, whether the components were heterogeneous *or* homogeneous.

- It focuses attention on the definition of *mutually understandable* message protocols so that the objects can communicate.

- It directs attention away from arbitrary syntactic differences among the components (e.g., among the query languages if the components are heterogeneous database systems).

As an *implementation approach*, the encapsulation provided by the object paradigm allows internal details of individual components to be concealed. Moreover, the object provides a framework for the incorporation of procedures required in interobject communication to be incorporated directly in the objects, such as procedures for data conversion. Also, the method inheritance concept provides a valuable facility for "homogenizing" heterogeneous components. The examples below illustrate these points.

Figures 2 and 3 show the steps involved in integrating heterogeneous DBMSs using an object-oriented approach. Figure 2 shows "objects" defined to encapsulate each of three entire heterogeneous DBMSs. The object encapsulation allows the operations provided by each DBMS to be invoked by messages from other objects. Clearly, nothing very exciting has happened so far with regard to integration; all that has been done is to establish communications among the components. Since the systems implement different data models, a given system will be unable to understand messages sent by the others. This illustrates the fact that thinking about, or implementing, components as objects is not an automatic solution to any problem. The *design* of the objects is an important consideration too.

Figure 3 illustrates the second step in the process, which involves defining (and implementing) object methods that implement messages whose semantics are mutually understandable by the various components. The approach is basically to surround the DBMS with a layer of software that implements a common interface, using the object concept to encapsulate this software. In this case, the common interface has been chosen to be relational, and to implement the relational **select** operation on all systems. This requires no translation on the relational system, but requires a translation method to be implemented on the other systems. (This "translation method" may, of course, be a very substantial piece of software if it is required to implement a complete relational interface on another form of DBMS!) A component playing the role of the translation method is frequently found in heterogeneous distributed DBMS architectures, such as that of CCA's MULTIBASE system [LAND82], where it is referred to as a *Local Data Interface* (LDI). The point is thus not to claim that the object-oriented approach provides entirely new capabilities in this case, but rather to illustrate that it provides a natural way to think about how they fit into an overall design approach, as well as to show how, as an implementation approach, object-orientation provides a

direct way of implementing these required capabilities.

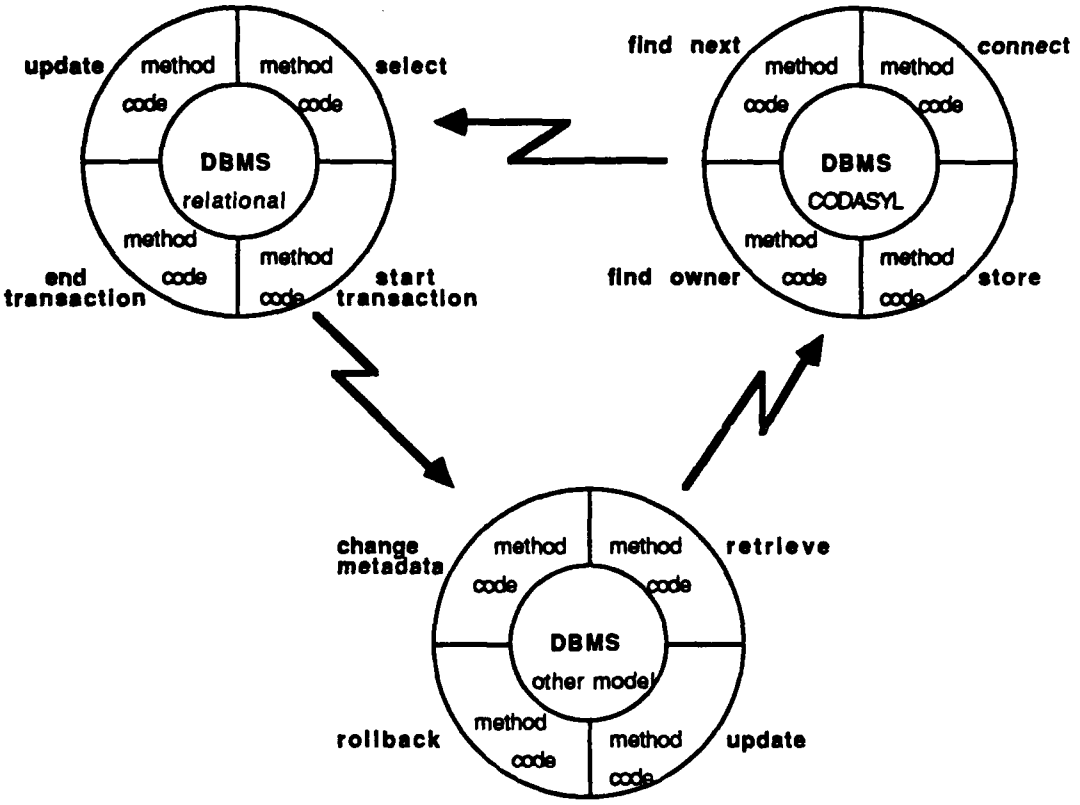


Figure 2 Heterogeneous Database Systems Encapsulated As Objects

Figure 4 shows the definition of objects at a smaller level of granularity. In this case, a part object class has been defined on each of two heterogeneous DBMSs. Note that while both classes implement a **part_number** method, the other methods are different. In the case of the **display** and **show** methods, the semantics are the same, but the names of the methods are different. In the case of the other methods, the semantics are different as well. This makes it difficult to integrate parts from the two systems. For example, without additional system facilities, the user would have to know what system a part came from in order to know how to display it (i.e., whether to issue the command **show** or **display**).

Figure 5 shows the use of the *inheritance* mechanism to define a generic type of part object class that permits integration of the two part object classes. The generic object class is defined as a supertype of the existing object classes, and as having methods that correspond to the methods possessed in common (at least semantically) by the two existing object classes. This technique of using inheritance to integrate heterogeneous object classes has also been used in CCA's MULTIBASE system [DAYA84], utilizing the inheritance capabilities of its DAPLEX data model.

Figure 6 shows the use of object encapsulation to implement the generic part object class defined in Figure 5. The translation method that implements the **part_number** operation on the generic part object class has only to dispatch the operation to the right underlying part, since this operation is supported by both underlying part object classes. However, the translation method that implements the **output** operation must determine whether the **display** or **show** operation should be sent to the underlying part.

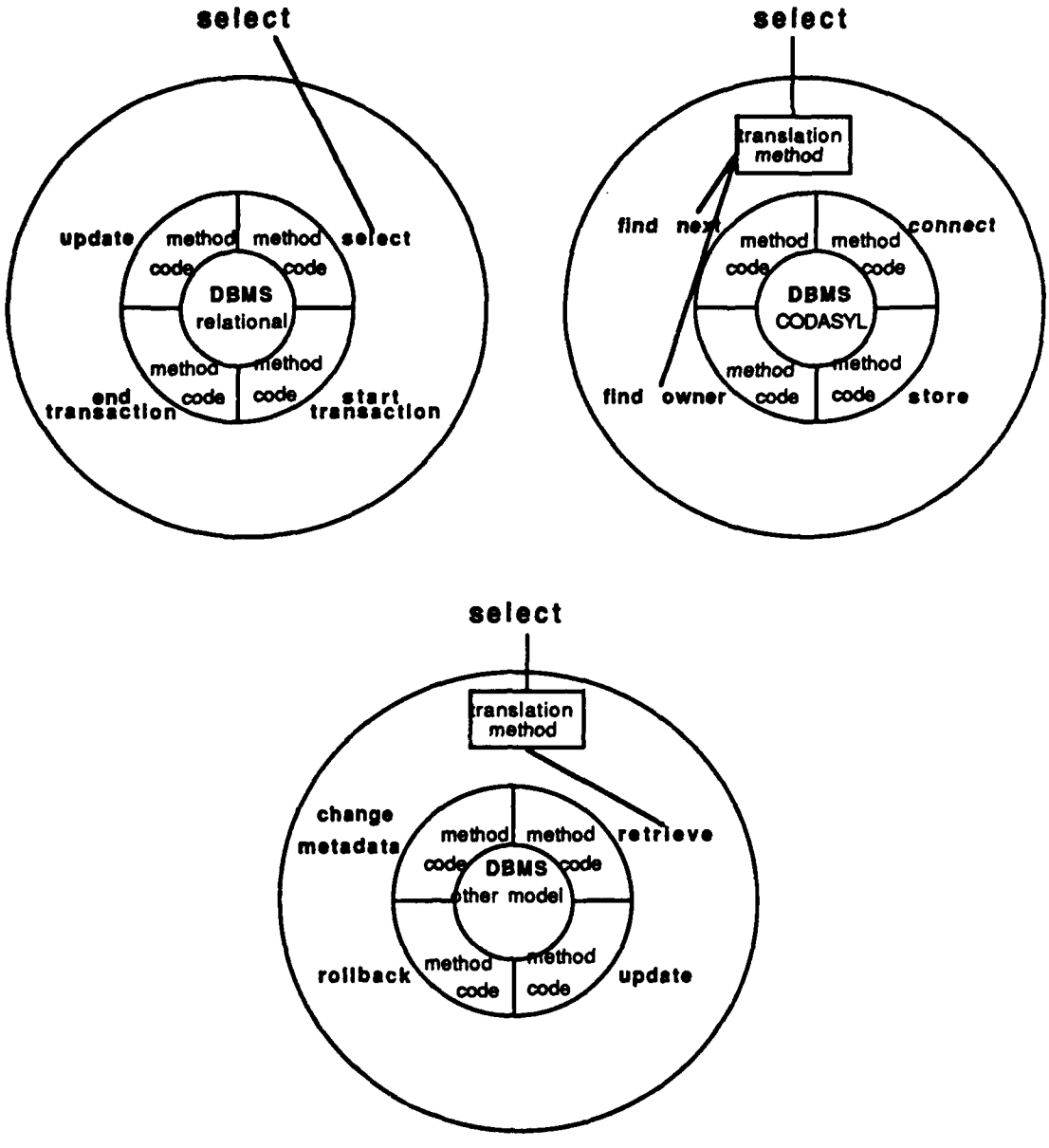


Figure 3 Integration of Heterogeneous DBMSs using Object Encapulation

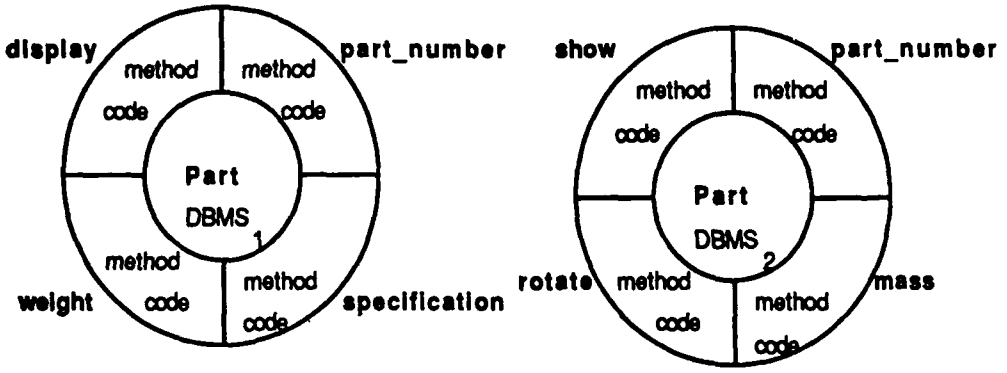


Figure 4 Part Object Classes on Two Heterogeneous Database Systems

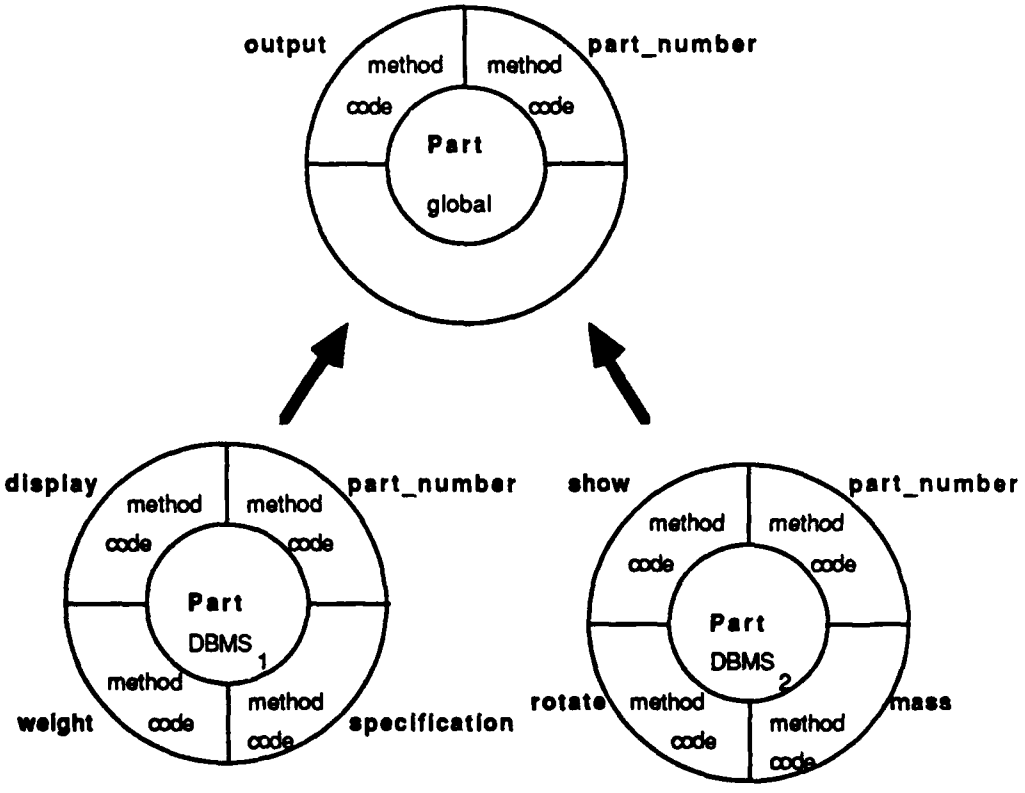


Figure 5 Definition of Generic **Part** Object Class Using Inheritance Concept

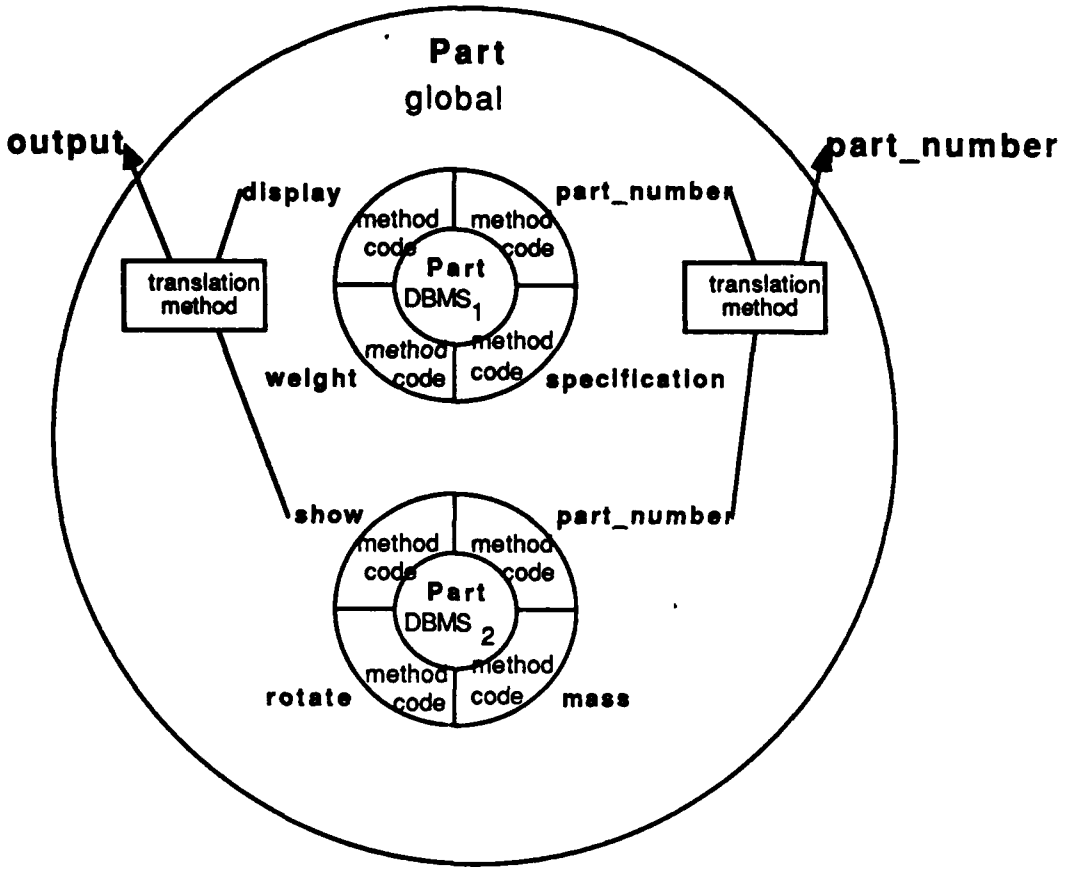


Figure 6 Integration of Part Object Classes Using Object Encapsulation

As in the previous example, the translation methods are effectively implementing what in conventional database technology would be called a *view*. Basically, a *view* consists of a set of data objects derived or transformed from stored data in the database, the purpose being to allow a user or application to "view" the data in a way that is more useful to it than the way the data was originally stored. Views are generally defined in database system using the facilities of the database's query language, but views defined in this way have a number of important limitations [KELL86]. Work such as [DAYA84] in heterogeneous distributed database technology has established that integration of these systems is basically similar to a view mapping problem. By allowing arbitrary procedures to be integrated into the model as object methods, the object-oriented approach provides a convenient way to provide increased view mapping capability. While some level of these facilities can be provided in more conventional approaches, it is often difficult and clumsy to do.

These examples illustrate that the object-oriented approach can be valuable in integrating heterogeneous systems (and data types), but is not a panacea. The conceptual advantages of object-oriented approaches must be backed up by implementation of facilities that support it. Morever, protocol standardization and rationalization is vital, since without this objects may not understand the messages directed to them, or be able to direct messages to other objects in appropriate ways. Users and designers of object-oriented systems will still require considerable interface support, since they must now understand a large set of objects in order to make use of the system. Finally, research and experience is needed in realizing the object-oriented approach in a heterogeneous environment. Current research on object-oriented database systems is addressing some of these issues, but little work exists that combines both elements of object-oriented research *and* heterogeneous distributed DBMS research.

Integration of Knowledge-Based Processing Using the Object-Oriented Approach

In integrated information systems, knowledge-based processing will be important in a number of ways:

- providing assistance to users and designers
- resolving differences in data semantics
- implementing advanced concurrency control mechanisms
- providing improved efficiency (through techniques such as "semantic query optimization")
- implementing application-specific rules
- versioning, configuration management, and security

Integrating this knowledge-based processing into a DBMS object model could be useful for a number of reasons. First, the generality, flexibility, and extensibility provided by the object-oriented approach should be extended to knowledge-based processing as well. For example, it is reasonable to expect that, over the system life cycle, existing knowledge-based components will have to be modified, and new knowledge-based components added. Moreover, these knowledge-based components may use heterogeneous knowledge representation techniques, either because the different techniques are more adapted to specialized requirements, or because advances in technology create new knowledge representation techniques that are included in new components. It seems likely that object-oriented techniques for the integration of heterogeneous components can also be used for the integration of heterogeneous knowledge-based components.

A second reason to integrate knowledge-based processing into the object model is that it enables the behavior of objects in the model to be derived from knowledge-based processing. In many cases, it may be irrelevant to outside objects whether the behavior of a given object is based on conventional processing, or on knowledge-based processing. The nature of the processing may thus be encapsulated within the object, and only the external behavior revealed. This provides a flexible way to integrate both conventional and knowledge-based components within the same system.

Finally, integration of knowledge-based processing into the object model allows the object-oriented DBMS to provide persistent object support for knowledge-based applications. It is anticipated that knowledge-based applications will increasingly require database support facilities, either because they are dealing with database-sized knowledge bases, or because the raw data they use as input to their processing is stored in databases so as to provide for access by other application programs. The coupling of knowledge-based and database systems is currently a very active area of research, and attempts have already been made to "loosely-couple" existing knowledge-based and database systems (by embedding database calls where database data is required). However, it is believed that performance considerations will demand a much tighter degree of integration of knowledge-based and database processing than can be achieved by such loose-coupling techniques (see, e.g., [JARK84]).

An example of research aimed at integrating knowledge-based processing with object-oriented database management is CCA's HiPAC project, which is just underway. The project is based on CCA's PROBE object-oriented DBMS described earlier. The intent is to enhance PROBE with a number of high-impact incremental facilities that increase support for knowledge-based applications. Specific enhancements being investigated include:

- Capturing heterogeneous knowledge-representations to enable knowledge-based applications to use PROBE as persistent storage.

- Support for condition monitoring over the database (the active DBMS paradigm). This enables a knowledge-based application to download situation-action (production) rules to the DBMS, so that the DBMS can notify the application of "interesting" situations in the database (a simple ver-

sion of this capability is described in [STON86a]). The action triggered by the condition becoming true could also involve more complex processing, including uploading of data from the database into an application's workspace, for knowledge-based analysis of the details of the situation.

- Representation of plans to support time-constrained scheduling. This is important to enable the DBMS to give priority to processing that might be extremely urgent, or to take compensating action when the time requirements of applications cannot be satisfied.

Work in these areas can build to some extent on existing work in multiparadigm knowledge programming systems, such as LOOPS [STEF86], which integrates conventional, object-oriented, and rule-based programming styles with condition monitoring (triggers, active values).

Future Directions

Object-oriented approaches create the potential for changes in the way data models evolve. Data models originally involved relatively few constructs. This forced users to do most of the work in mapping from the real world they wanted to model to the constructs provided by the data model. A more recent trend in data model development is to define more and more special-purpose constructs directly in the data model. Recent examples have been various proposals for "version" constructs (for use in engineering information systems), and various forms of space and time data types. The advantage of defining such constructs in the data model is that their special semantics can be directly captured in the DBMS implementation, thus making it possible to optimize processing of requests involving those data types. However, while such constructs provide extended, tailored facilities to the users of the data model, these constructs are fixed by the data model designer. Thus, they may not exactly capture the semantics required by the particular application involved, and in any event cannot adapt to changing requirements. The object-oriented approach provides the best of both worlds, since new constructs can be user-defined, and thus highly tailored to particular application requirements, but the new constructs can behave as if they were "part of the data model".

Object-oriented approaches also identify required data model research directions. For example, to take full advantage of the ability to support user-defined types, required implementation support, such as extensible query optimization facilities, must be present in the underlying system. Methods of defining procedural information within object-oriented data models also require investigation. The ability to define procedures in conventional programming languages and invoke them from the object model is one approach, but in general we expect "tighter coupling" of procedural and conventional DBMS capabilities, so as to support query optimization and concurrency. Adding persistent objects to an object-oriented programming language, as in the GEMSTONE DBMS [MAIE86], is a possible approach. However, this does not address the fact that DBMSs conventionally support multiple programming languages. Architecturally, this raises the issue of where one draws the line between "database system" and "application program" in such a framework (or whether such lines are still appropriate). Optimization to achieve good performance in such architectures is a challenge, since it involves closely integrating DBMS and programming language forms, smart staging over a large "object memory", and dealing with complex and possibly large objects. Ultimately, this may lead to new organizations of "operating system", "programming language", and "DBMS" facilities, in order to support these advanced requirements.

Acknowledgements

The author wishes to thank Michael Brodie, Umesh Dayal, and the other members of CCA's PROBE project, for their contributions to this paper.

References

[CARE86] M. J. Carey, et. al., "The Architecture of the EXODUS Extensible DBMS", in [DITT86], 52-65.

[COX86] B. J. Cox, *Object Oriented Programming*, Reading, Addison-Wesley, 1986.

[DAYA84] U. Dayal and H. Hwang, "View Definition and Generalization for Database Integration in a Multidatabase System", *IEEE Trans. Software Engineering,* 10(6), 628-645, 1984.

[DAYA86] U. Dayal and J. M. Smith, "PROBE: A Knowledge-Oriented Database Management System", in M.L Brodie and J. Mylopoulos (eds.), *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies*, New York, Springer-Verlag, 1986.

[DITT86] K. Dittrich and U. Dayal (eds), *Proc. Intl. Workshop on Object-Oriented Database Systems,* Washington, IEEE Computer Society Press, 1986.

[HUDS86] S. E. Hudson and R. King, "CACTIS: A Database System for Specifying Functionally-Defined Data", in [DITT86], 26-37.

[JARK84] M. Jarke, J. Clifford, and Y. Vassiliou, "An Optimizing Prolog Front-End to a Relational Query System", *Proc. SIGMOD '84,* Boston, MA, June 1984.

[KELL86] A. Keller, "Choosing a View Update Translator by Dialog at View Definition Time", *Proc. Twelfth Intl. Conf. on Very Large Data Bases,* Kyoto, Japan, August 1986.

[LAND82] T. Landers and R. L. Rosenberg, "An Overview of MULTIBASE", in H. J. Schneider, ed., *Proc. 2nd Intl. Symp. on Distributed Databases*, Berlin, W. Germany, September 1982.

[LOCH85] F. Lochovsky (ed.), *Database Engineering*, Vol. 8, No. 4, Special Issue on Object-Oriented Systems, IEEE Computer Society, 1985.

[MANO86a] F. Manola and J. Orenstein, "Toward a General Spatial Data Model for an Object-Oriented DBMS", *Proc. 12th Intl. Conf. on Very Large Databases*, 1986.

[MANO86b] F. Manola and U. Dayal, "PDM: An Object-Oriented Data Model", in [DITT86], 18-25.

[MAIE86] D. Maier, J. Stein, A. Otis, and A. Purdy, "Development of an Object-Oriented DBMS", *Proc. ACM Conf. on Object Oriented Programming Systems, Languages, and Applications*, Portland, Oregon, September 1986..

[ROSE86] A. Rosenthal, et.al., "Traversal Recursion: A Practical Approach to Supporting Recursive Applications", *Proc. ACM-SIGMOD Intl. Conf. on Mgmt. of Data*, 1986.

[SHIP81] D. Shipman, "The Functional Data Model and the Data Language DAPLEX", *ACM Trans. Database Systems,* 6(1), 140-173.

[SKAR86] A. H. Skarra, S. B. Zdonik, and S. P. Reiss, "An Object Server for an Object-Oriented Database System", in [DITT86], 196-204.

[STEF86] M. J. Stefik, D. G. Bobrow, and K. M. Kahn, "Integrating Access-Oriented Programming into a Multiparadigm Environment", *IEEE Software,* 3(1), January 1986.

[STON86a] M. Stonebraker, "Triggers and Inference in Database Systems", in M. L. Brodie and J. Mylopoulos, eds., *On Knowledge Base Management Systems*, Springer-Verlag, New York, 1986, 297-314.

[STON86b] M. Stonebraker, "Object Management in POSTGRES Using Procedures", in [DITT86], 66-72.

A DISTRIBUTED DATABASE ARCHITECTURE FOR AN INTEGRATED MANUFACTURING FACILITY

Vishu Krishnamurthy, Y.W. Su
Herman Lam

University of Florida

Mary Mitchell, Ed Barkmeyer

National Bureau of Standards

Contributed by Mary Mitchell

1. INTRODUCTION

Computer Integrated Manufacturing (CIM) includes activities for supporting, planning, and controlling the manufacture of products. The objective of CIM is to improve the acquisition and delivery of manufacturing information thus improving manufacturing effectiveness and resource utilization. Computer aided systems do exist at this time. However, most of these systems and methods have been designed and used independently to support specific CIM functions. The result is a limited interaction among the different component functions of CIM. The totally integrated manufacturing system is still non-existent.

A survey of CAM [HAT83] and a more recent discussion on Engineering Information Systems [GAD87], stresses that software integration is the technology in CIM with the greatest potential to improve productivity. To realize the full benefits of CIM, the fully integrated manufacturing system must 1) link the activities on the shop floor with the design, factory planning and manufacturing engineering functions, 2) provide accurate and timely feedback from the factory floor, and 3) modify and replan activities based on actual performance feedback.

The design of a system to support the information requirements of automated manufacturing [WED87] is affected by the environment in which it operates. The general requirements applicable to any CIM system are the following.

a) Integration of Heterogeneous Systems:

The use of different systems for CAD having different database management systems is not uncommon. Manufacturing components such as robots, machine tools, vision systems, measurement systems, etc. are produced by different vendors and have diverse control protocols, capabilities, and interface protocols.

b) Flexible Manufacturing:

A CIM system should provide the ability : 1) to adjust for future production demands, 2) to meet dynamically changing states in the production environment, and 3) to expand with some degree of modularity. It should also provide the capability to logically or even physically regroup the equipment or reassign resources to meet changing production requirements.

c) Local Autonomy and Integrated Operation:

In the development stage of the integerated facility, whenever a new equipment is introduced to the factory data network, the factory control must provide for independent testing and integration. The data must remain consistent and an independent set of data and data manipulation capabilities must be provided for the subsystem.

d) Time Critical Operations:

Many systems that direct equipments on the factory floor are real time control systems, which make use of sensory data to perform their functions. In an automated environment, these sensory and other forms of feedback are data resources that need to be shared. Real-time control systems imply that delays imposed by data sharing and communications will not be tolerated.

e) Adaptive Control:

The control systems in an automated factory are able to make use of the sensory inputs and other feedbacks to intelligently react to failures, discrepancies, and unexpected events.

It is evident from the CIM requirements listed above that data sharing is a principal requirement in a CIM environment. Therefore, central to a fully integrated manufacturing system is a data administration system that manages the shared data resources throughout the design, planning and manufacturing processes [SMI86]. Other manufacturing integration efforts such as the NASA's IPAD and the Air Force's ICAM projects [FUL80, ICA79, ICA83] have recognized the importance of data management in CIM by concentrating on the study and development of distributed database management techniques. The issues of representation, integration [APP85] and administration of CAD/CAM data have been identified as some of the major problems in CIM [YOS81].

This paper focuses on the design of a distributed database management system architecture to support integrated manufacturing. A prototype system has been implemented and is being used in the National Bureau of Standards' Automated Manufacturing Research Facility (AMRF). The AMRF is being developed as a testbed for automated small batch manufacturing [ALB81, SIM82, MCL83, NAN84], in an effort to examine the requirements and features of CIM in more detail and to identify measurements and standards needs for CIM.

The paper is organized as follows. Section 2 describes the overall architecture of the Integrated Manufacturing Data Administrative System (IMDAS). Sections 3 and 4 briefly describe the global data model (OSAM*) and the global data manipulation language designed and prototyped for the distributed database system, respectively. Section 5 then details the various modules of the IMDAS. In section 6, we present a processing scenario of the IMDAS. Finally, in Section 7, we conclude with some notes on our future efforts in this project.

2. THE ARCHITECTURE OF THE INTEGRATED MANUFACTURING DATABASE ADMINISTRATION SYSTEM (IMDAS)

IMDAS supports an integrated database which is physically distributed over a number of component systems (see Figure 1). In order to map from the logically integrated database to physically distributed databases, IMDAS provides three levels of view definition. They are: 1) global external views, 2) global conceptual view, and 3) fragmented views. A global external view defines a portion of the integrated database as seen by a single user, typically a control process. It identifies the data objects and their associations useful to the user. The global conceptual view represents the integrated database and is a comprised view of all enterprise data required to manage the CIM. The fragmented views represent the physical partitioning or replication of the conceptual objects across the component systems. Each fragmented view represents the data objects residing on a component system.

2.1. CONSIDERATIONS IN ARCHITECTURE SELECTION

Distributed database architecture [CERI84] falls into two basic categories: centralized control and distributed control. In the centralized architecture, a single central data administration system handles all the control and management functions. Data is stored in distributed databases, but all database requests are routed to a central control module. Although the architecture is straightforward to develop and easy to manage, it suffers from the lack of autonomous operation, predictable service times for time critical operations, and adaptive recovery and reliability due to single point of control - all of which are critical issues in CIM.

In a fully distributed architecture, every component system has a data administration system that can control and manage distributed execution of database requests originated by its own processes and is

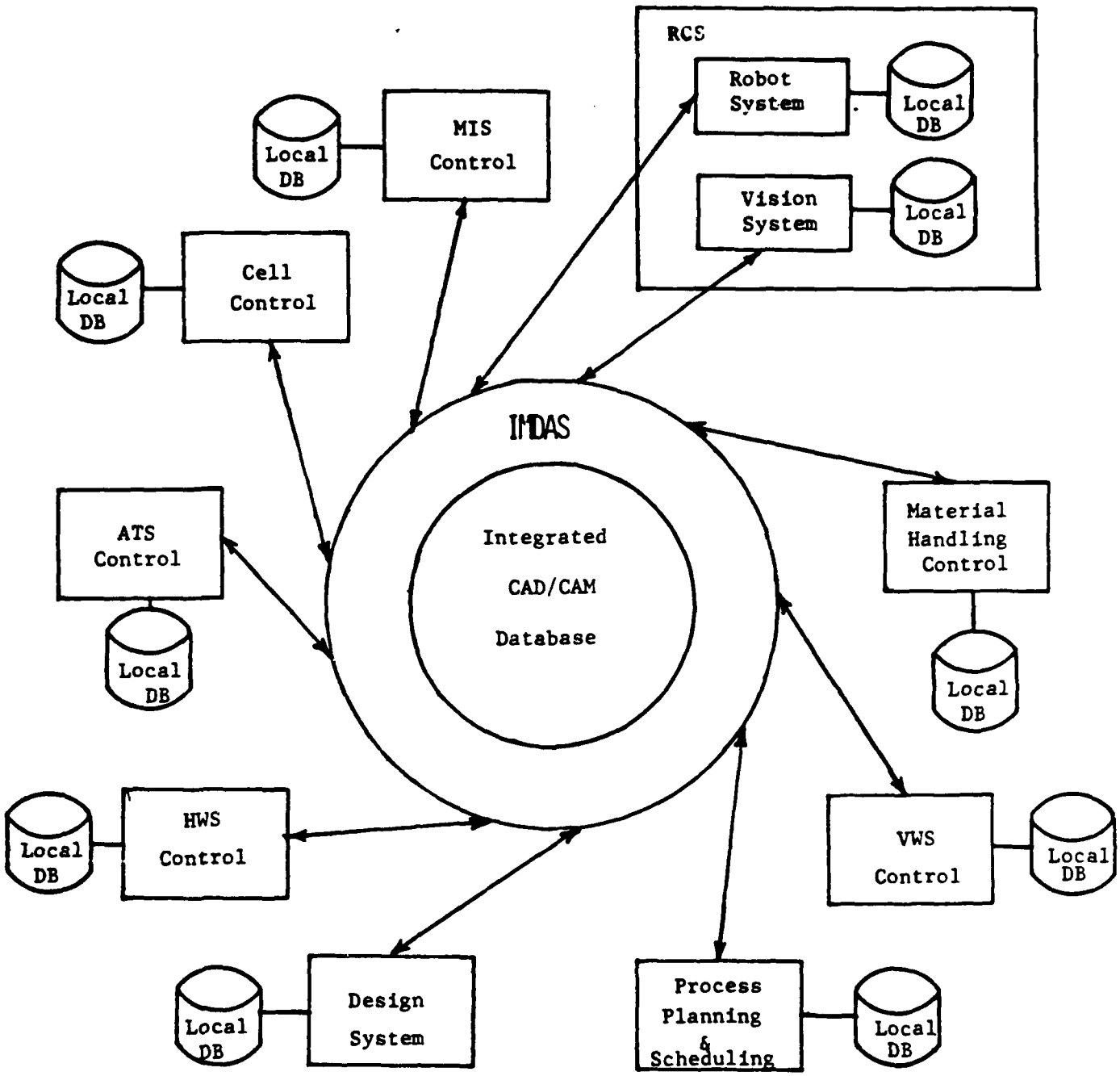


Figure 1. The Integration of Heterogeneous Component Systems in a Factory Network

also capable of servicing requests coming from other component systems. This architecture makes the system more reliable and provides each component system with maximum autonomy. Unfortunately, fully distributed control of database integrity, concurrency and recovery is far more difficult to achieve than a centralized control. Most fully distributed database systems manage homogeneous databases on similar component systems. In contrast, a CIM system generally contains heterogenous databases that reside in dissimilar component systems. Therefore, the considerations that led to the selection and use of distributed control for homogeneous distributed database management systems such as Distributed INGRES, System R*, SDD-1, and POREL [WIL82, NEU84, CER84, BRA82] are not necessarily valid for the CIM environment. Moreover, implementing a totally distributed database software on every heterogenous component system in a CIM environment is not as feasible, technically or economically, as implementing it on many homogeneous systems. A significant overhead is imposed in a fully distributed system by the handshaking required by the distributed control, and this has a negative impact on the performance of real-time operations.

The IMDAS design pursues a third possibility [BAR86, SU86]: a hybrid architecture which has the features of both centralized and distributed controls, thus possessing the advantages of both control strategies and, at the same time, avoiding some of their individual drawbacks. The IMDAS consists of three levels of control, each of which is responsible for a defined set of distributed database management functions. These functions are distributed over the CIM component systems according to their computational capabilities. The different levels of control cooperate to establish, manipulate, and control the distributed databases.

2.2. THE THREE LEVEL ARCHITECTURE OF IMDAS

The IMDAS provides each component system with one or more classes of data management services. These classes are referred to as basic, distributed and master services.

The basic data management services include command translation, data translation and network communication. They are required by the control processes residing on every component system. These services are provided by a Basic Data Administration System (BDAS), which is implemented and installed on every component system.

The distributed database control and management functions include query translation, distributed query execution, and final data assembly. They are handled by the Distributed Data Administration System (DDAS). This software resides on those component systems that are powerful enough to support it. Not every component system has a DDAS, but each has access to one.

The "master" class of data management services are required when different component systems or sub-groups of them (some containing more than one DDAS) are integrated into an operating segement of the factory data network. For each independently operating segment of the factory, there is exactly one Master Data Administration System (MDAS), which carries out such services as network initialization, managing the master data dictionary, and resolving concurrency problems between DDASs. The MDAS resides on one of the DDAS component systems.

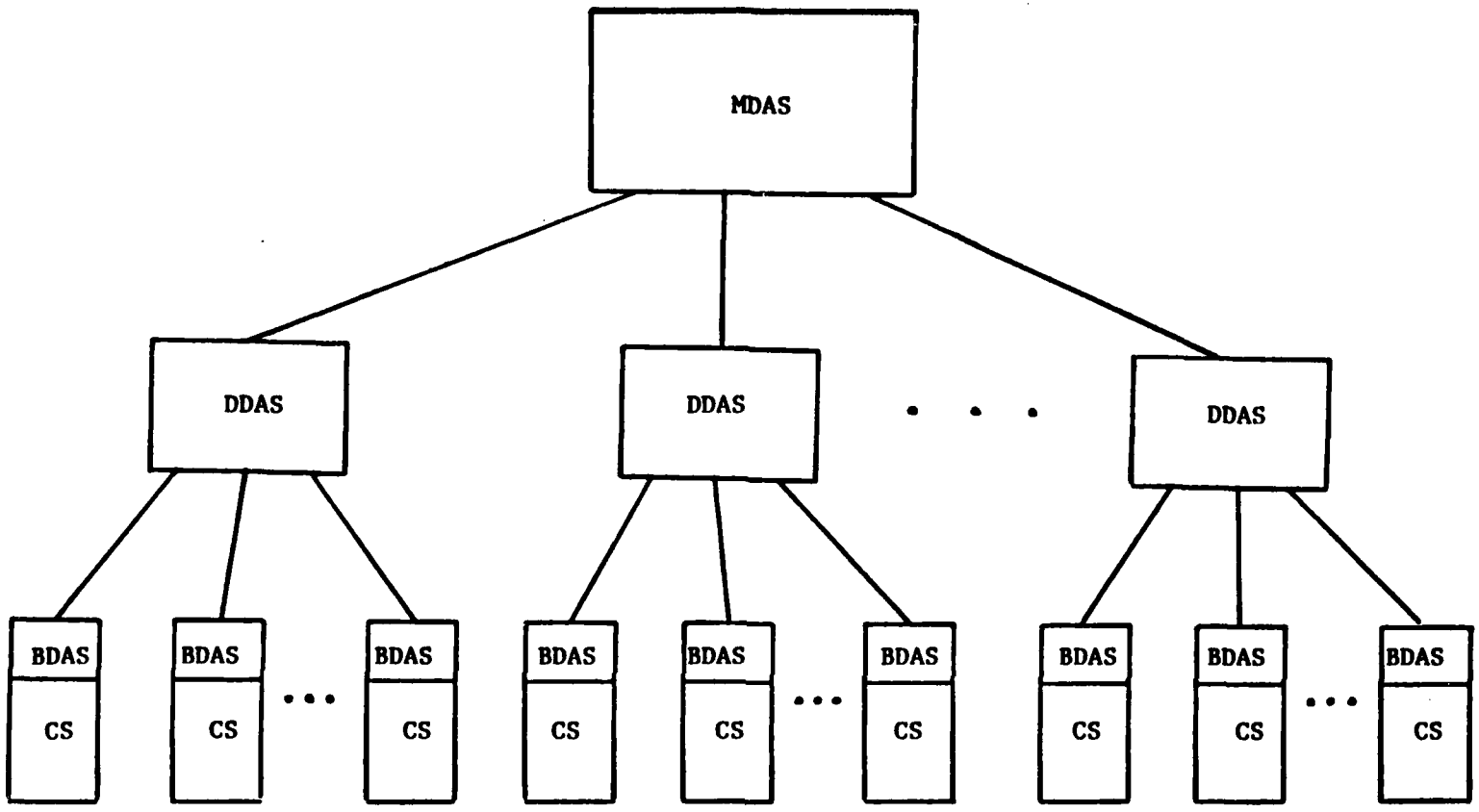
A three level hierarchy exists among the BDAS, DDAS and MDAS modules of the IMDAS (see Figure 2). As illustrated in Figure 1, the integrated database is physically distributed on different component systems, and the BDAS, DDAS, and MDAS modules serve as a central core through which all the CIM subsytems interact and communicate.

3. A GLOBAL DATA MODEL FOR IMDAS

Within the various levels of the AMRF control hierarchy, the control processes access the integrated database through the IMDAS. For the purpose of providing a uniform interface to the integrated database, the integrated database is modeled using a common global data model, and the database requests are made using a common global DML. In this project, the common global data model used is the Semantic Association Model (SAM*) [SU83, SU85] and its more recent extension to an Object-oriented SAM* or OSAM* [SU87].

The OSAM* models objects found in an integrated CAD/CAM database in terms of complex data structures, rules and constraints. An OSAM* database is a collection of object classes. Each object class consists of an object-type definition and its instances. An OSAM* class is defined by the following:

(1) Name - This is the name of the class or the object-type it represents;



MDAS - Master Data Admininstration System

DDAS - Destributive Data Administration System

BDAS - Basic Data Administration System

CS - Component System

Figure 2. The Three-Level Architecture of IMDAS

(2) Structure - This is defined by the associations of the class to other classes. The various possible associations[1] are Membership (M), Aggregation (A), Interaction (I), Composition (C), Cross-product(X), Generalization (G), and data type constructors such as **Set, Bag, Ordered-set, Vector, Matrix,** etc.;

(3) Operations - These are all the valid operations (or methods) that are associated with the class or the instances of the class;

(4) Knowledge Rules - These are general inference rules, expert rules, and semantic constraints associated with the class or the instances of the class.

The OSAM* model recognizes two different types of classes: (1) *entity object* class and (2) *domain object* class. An *entity object* class represents a class of instances, which are entities in an application domain and are, therefore, explicitly created by the user of the database. A *domain object* class is a class that strictly represents a domain of data values for some attribute used to describe an object in the database.

A detailed description of the OSAM* data model is beyond the scope of this paper. For a detailed discussion of OSAM*, refer to [SU83, SU85, SU87]. In this paper, we shall briefly describe some of the features of OSAM* and illustrate some of its capabilities with an example.

The OSAM* model provides, besides the schema, two additional representations of the database : (1) **Semantic diagram (S-diagram)** and (2) **Generalized relation (G-relation)**. The **S-diagram** depicts the structure and some of the constraints of the object classes in the database by means of a network of labeled nodes and links. For example, the S-diagram in Figure 3 illustrates a portion of an integrated manufacturing database, which describes items, trays, parts, devices, containers, and storage locations and the relationship that a storage location may contain an item. In the S-diagram, nodes stand for object classes. A rectangular node represents an entity object class, while a circular node represents a domain object class. Links emanating from a node represents the structural attributes of the class. A link drawn from class A to class B stands for an association of an object in A to some object in B. The type of associa-

[1]Note that the Summarization Association type discussed in [SU83, SU85] has been dropped and its semantics are incorporated in the descriptive attributes associated with a class as described in [SU87].

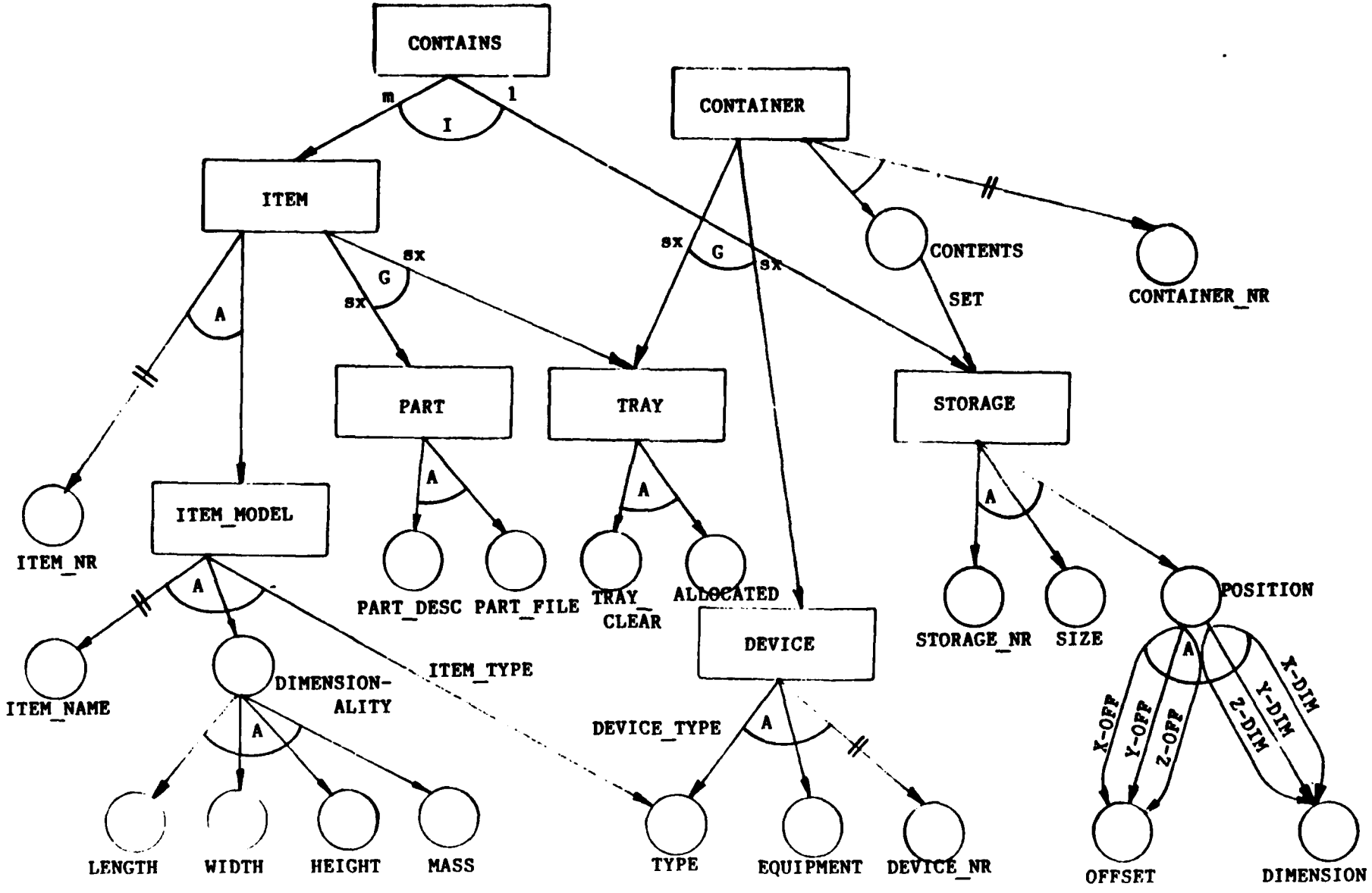


Figure 3. Schema of a Portion of an Integrated Manufacturing Database

tion is indicated by labeling the link as one of the following: M, A, I, C, X, G, set, bag, oset, vector, or matrix. An object class is defined by its associations with other object classes. The associations are represented by labeled attributes. Each attribute is assigned a name, which can be the same as the object class from which it draws its members (i.e., the domain). For example, the attribute DEVICE_NR of the class DEVICE (i.e., the link) has the same name as its domain DEVICE_NR. The name of an attribute can be different from that of its domain. In this case, the link representing the attribute is explicitly named in the S-diagram (e.g., the attribute DEVICE_TYPE of class DEVICE). In order to uniquely identify an instance of a class, an attribute whose values uniquely identify the instances of the class, is designated as the object identifier. The object identifier is indicated in the S-diagram by an attribute link with double dash-marks across it. For example, DEVICE_NR of class DEVICE is an object identifier.

The example shown in Figure 3 illustrates five types of OSAM* associations: Membership, Aggregation, Interaction, Generalization, and Set. The Membership association type (not shown in the diagram) associates objects of a domain object class with some primitive domain object class such as a STRING, INTEGER, REAL etc.. These primitive domain object classes contain atomic data values and are system-predefined. For example, in Figure 3, the object class ITEM_NAME is a domain of item names of data type STRING.

The Aggregation association type models the semantics that each object of a class is described or characterized by an aggregation of objects of some other classes. In the S-diagram, the attribute links of a class, which point to the class's constituent classes, are grouped together by an arc designated by the letter A for aggregation. For example, in Figure 3, an instance of the class STORAGE is defined by an aggregate of objects drawn from the classes STORAGE_NR, STORAGE_SIZE, and POSITION.

An interaction between or among a set of objects is itself an object. An object class, which represents a set of interactions, is said to enter an interaction association with its participant object classes. In the S-diagram, the participant object classes of an interaction are connected to the defined class by attribute links, which are grouped together by an arc designated by the letter I for interaction. The cardinality constraints existing between participant classes are depicted on the arc. In Figure 3, CONTAINS is an object class with an interaction association with the classes ITEM and STORAGE. The class CONTAINS models

ITEM

OID : G	ITEM_NR : A	ITEM_MODEL : A
		OID
05	part_1	012
06	part_2	014
07	part_3	012
08	part_4	014
09	tray_1	013
010	tray_2	013
011	tray_3	013

PART

OID	PART_DESC : A	PART_FILE : A
05	blank	PF_1
06	T_joint	PF_2
07	L_joint	PF_3
08	crank	PF_4

TRAY

OID	TRAY_CLEAR : A	ALLOCATED : A
09	6	yes
010	4	no
011	10	no

Figure 4. G-Relations for Entity Object Classes shown in Figure 3.

a class of facts that some storages contain some items.

A class may contain generic properties of objects, while another class may contain more specialized properties of those objects. The class, which contains generic properties, is called a superclass and the class, which contains more specialized properties, is called a subclass. A superclass may have a number of subclasses. The subclasses are connected to their superclass by attribute links in a S-diagram. These attribute links are explicitly labeled by the letter G for Generalization. An object in the subclass also belongs to its superclass, and thereby it "inherits" all the properties (i.e. attributes, operations and rules) of the superclass. The example in Figure 3 contains two generalization associations among object classes. First, the object classes PART and TRAY are subclasses of the class ITEM. This indicates that some subset of items are trays or parts. The set exclusion (SX) constraint between the classes TRAY and PART (shown flanking the arc between the structural attributes in the figure) precludes any tray from being a part and vice-versa. Second, the superclass CONTAINER contains DEVICE and TRAY as its subclasses. The generalization association models the fact that a device or a tray is also a container. The generic properties of trays and devices are captured in the class CONTAINER.

The set association between two object classes A and B is modeled in an S-diagram by an attribute link connecting A to B. The link is explicitly labeled by SET for set association. Each instance of A is a set of instances of B. The set association is a data type constructor, which constructs sets out of the domain B. For example, in Figure 3, the class CONTENTS represent a class whose instances are sets formed from the instances of the class STORAGE.

We now turn our attention to the other representation of a database. A **G-relation** is a tabular representation of the instances of an object class. The first row of a G-relation contains the object class attributes and the rest of the rows represent the object instances. A G-relation is defined recursively and results in a nested tabular structure. In other words, it is a relation (in the relational model sense) in which the attributes themselves can be relations. It has a set of distinct attributes: the object identifier, which is distinguished from the others by an underlined attribute name, and structural attributes, which are explicitly labeled by their association types. The G-relation is the standard format in which the result to a user's query is presented. The data associated with the instances of classes are stored internally as G-

relations, which represent the logical storage structure of the instances of a class.

Figure 4 shows the G-relational representation of the example in Figure 3. Every entity object class in the S-diagram has a G-relation representation. Every entity object class instance is assigned by the system a unique object identifier which is referred to as OID. An object class instance is referred to by the instances of other object classes through its OID. For example, in the G-relation representation of ITEM, the attribute ITEM_MODEL contains the OIDs of the instances stored in the G-relation of the object class ITEM_MODEL. In processing a query, the user specified DML is converted into a query tree consisting of operations on G-relations.

4. A GLOBAL DATA MANIPULATION LANGUAGE FOR IMDAS

Database requests in the IMDAS are posed in a common, global data manipulation language (GDML). The GDML is modeled after SQL [ANS86], which is a high level, non-procedural, set oriented language for relational databases. Features of the GDML include simple structures, powerful operators, short initial learning period, improved data independence, and integrated data and metadata manipulation. Additional features of the GDML for supporting CAD/CAM applications are:

(1) High level specification of the database request; the user need not specify the traversal path of the query since it is already represented by the semantics of the database.

(2) Support for complex and abstract data types, which facilitates the modelling of engineering/manufacturing data.

(3) Strong data type checking for complex and abstract data types.

(4) Provision for the expression and execution of knowledge rules, thus providing the infra-structure for knowledge base management.

(5) Automatic enforcement of constraints. For example, based on the semantics specified in the OSAM* schema, an update requests can trigger additional database operations automatically to maintain the consistency in the database.

ITEM_MODEL

OID	ITEM_NAME : A	ITEM_TYPE : A	DIMENSIONALITY : A			
			LENGTH : A	WIDTH : A	HEIGHT : A	MASS : A
012	BLANK	PART	10	5	3	15
013	TRAY	TRAY	20	10	10	40
014	BLANK_1	PART	15	10	5	30
015	GRIPPER	R_GRIP	20	5	3	15

CONTAINER

OID : G	CONTAINER_NR : A	CONTENTS : A
		STORAGE : SET
		OID
01	cart_1	015 016 017
02	unit_1	018 019
09	tray_1	020
010	tray_2	021
011	tray_3	022
03	end_effec	023

DEVICE

OID	DEVICE_NR A	EQUIPMENT : A	TYPE : A
01	cart_1	V_MTOOL	cart
02	unit_1	H_TOOL	unit
03	end_effec	V_ROBOT	end_effector

Figure 4. (continued)

(6) Support the attribute inheritance property of the OSAM* generalization association.

A query is posed by viewing the S-diagram. The language allows the user to formulate a query by naming the objects and attributes of his/her interest. Since the inter-relationships among objects are already explicitly represented in the database, no explicit joins need be specified in the GDML. This is unlike the relational SQL in which the effect of joins is achieved by nesting SELECT_FROM_WHERE blocks. All that is necessary in the GDML, is the specification of the object classes, about which information (attributes and values) is to be retrieved, along with the condition of retrieval. The attributes and values to be retrieved can either be those of the class itself or can be derived through the associations between the object class and other object classes. A detailed discussion on the GDML is beyond the scope of this paper. Interested readers may refer to [KRI85, KRI87]. We shall illustrate some of the capabilities of the GDML with an example:

QUERY:

```
SELECT      device_nr, equipment,
            (SELECT storage_nr, (item_nr IN CONTEXT OF Contains)
             FROM   Contents
             WHERE size = 10)
FROM        Device
WHERE       device_nr = 'cart_1'
USE MEMORY  Device_contents
```

The above query is based on the S-diagram shown in Figure 3. Its intent is to retrieve the device_nr and the equipment of device 'cart_1', and, for that device, retrieve the storage_nr of its storage whose size equals 10, and the item_nr of the item contained in that storage location. The result of this query is presented in a G-relational form shown in Figure 5. Some of the features of the GDML illustrated in this query are :

(1) Transparent paths: If there is no ambiguity present in the S-diagram, i.e. there are no multiple paths between them, it is not required that a complete path from a class or a complex attribute to its nested attributes be specified in the GDML, For example, in the given query, the path from CON-

STORAGE

OID	STORAGE_NR : A	POSITION : A						SIZE : A
		X_OFF	Y_OFF	Z_OFF	X_DIM	Y_DIM	Z_DIM	
015	S0	*	*	*	*	*	*	10
016	S1	*	*	*	*	*	*	20
017	S2	*	*	*	*	*	*	10
018	S3	*	*	*	*	*	*	20
019	S4	*	*	*	*	*	*	20
020	S5	*	*	*	*	*	*	10
021	S6	*	*	*	*	*	*	10
022	S7	*	*	*	*	*	*	10
023	S8	*	*	*	*	*	*	20

Figure 4. (continued)

CONTAINS

OID	ITEM : I	STORAGE : I
	OID	OID
031	05	015
032	09	019
033	07	015
034	06	020
035	011	021

Figure 4. (continued)

RESULT

DEVICE_NR : A	EQUIPMENT : A	CONTENTS : A	
		STORAGE : SET	
		STORAGE_NR : A	CONTAINS : A
			ITEM : SET
			ITEM_NR : A
cart_1	V_MTool	S0	part_1
			part_3
		S2	-

Figure 5. Result of the Example Query

TENTS to attributes SIZE is derived from the inter-connections of the S-diagram of Figure 3. Since the semantics of the database has been defined in the schema, the user should not have to restate it in the query.

(2) Property Inheritance: CONTENTS is an attribute defined for the CONTAINER. Since a DEVICE is a CONATINER, this property is inherited by DEVICE.

(3) Complex projection list: Further database operations can be performed on non-atomic attributes (e.g. CONTENTS) in the projection list of the query.

(4) Data delivery specifications: In a CIM application, most of the queries to the database are issued by application programs. Hence, the data retrieved from the database must be delivered to some location that is easily accessible by these programs. In AMRF, two such delivery locations are possible: the common memory[2] and the disk. In the example query, the **USE MEMORY** clause indicates that the data is to be delivered to a common memory location named by the user as Device_contents. If the results are to be delivered to a disk file, then the **USE FILE** clause is used. In general, the USE clause can be expanded to specify the format of the result, and the delivery mode of the data such as simple, stream, or block.

5. FUNCTIONAL DESCRIPTION OF THE IMDAS MODULES

The IMDAS architecture's hierarchical structure (Figure 2) has no direct relationship to its control topology. There are three levels of software which form the fully functional distributed IMDAS. These are the Basic Data Administration System (BDAS), the Distributed Data Administration System (DDAS), and the Master Data Administration System (MDAS).

5.1. BASIC DATA ADMINISTRATION SYSTEM (BDAS)

The BDAS provides a uniform interface between the data residing on each component system and the integrated database. Since the component computer systems have widely different processing capabilities, the BDAS provides only the essential data management and communication functions. These functions

[2]A shared memory scheme [MIT84], in which a process stores information of a particular type for a particular type of recipient

may be implemented differently on each component system in order to cater to the idiosyncracies of that system. The different functional modules of the BDAS illustrated in Figure 6 are: a) Basic Service Executive (BSE), b) Data Base Management or File Management, c) Report Generator, d) Command Translation (CT), e) Data Translation (DT), f) Interprocess Communication, Common-memory (CM), and g) Network Communication.

a) Basic Service Executive (BSE)

The function of this element is to provide a single point of contact between the DDAS and BDAS data services. The BSE accepts commands from DDAS and routes them to the appropriate BDAS modules, which perform the command translation, data manipulation, and data translation functions. It also constructs data delivery paths, sequences the execution of query tree operations, and reports completions and problems back to the DDAS transaction manager.

b) Data Base Management or File Management

The BDAS on each component system must provide, through the local database or file management system, the actual access to and manipulation of the local databases. The database manipulation and management capabilities vary substantially from system to system, depending on the facilities provided by the software. For example, in small systems, the local database may be implemented only in the main memory and managed by a shared memory management process; in others, a file manager may meet all local requirements, giving the system little more than "read" and "write" operations. In larger systems, sophisticated database management systems may provide the actual database services. Moreover, there may be more than one such facility on a single component system. The data manager must also be capable of cooperating in distributed consistency, concurrency, and recovery control.

c) Command Translation (CT)

In IMDAS, a query issued in the Global Manipulation Language (GDML) by a component system is translated into a tree of primitive operations by a DDAS. The DDAS then sends these primitive operations to the appropriate BDASs. The BDAS then uses the Command Translator to convert the received primitive operations into a sequence of operations in the language of the local data or file manager.

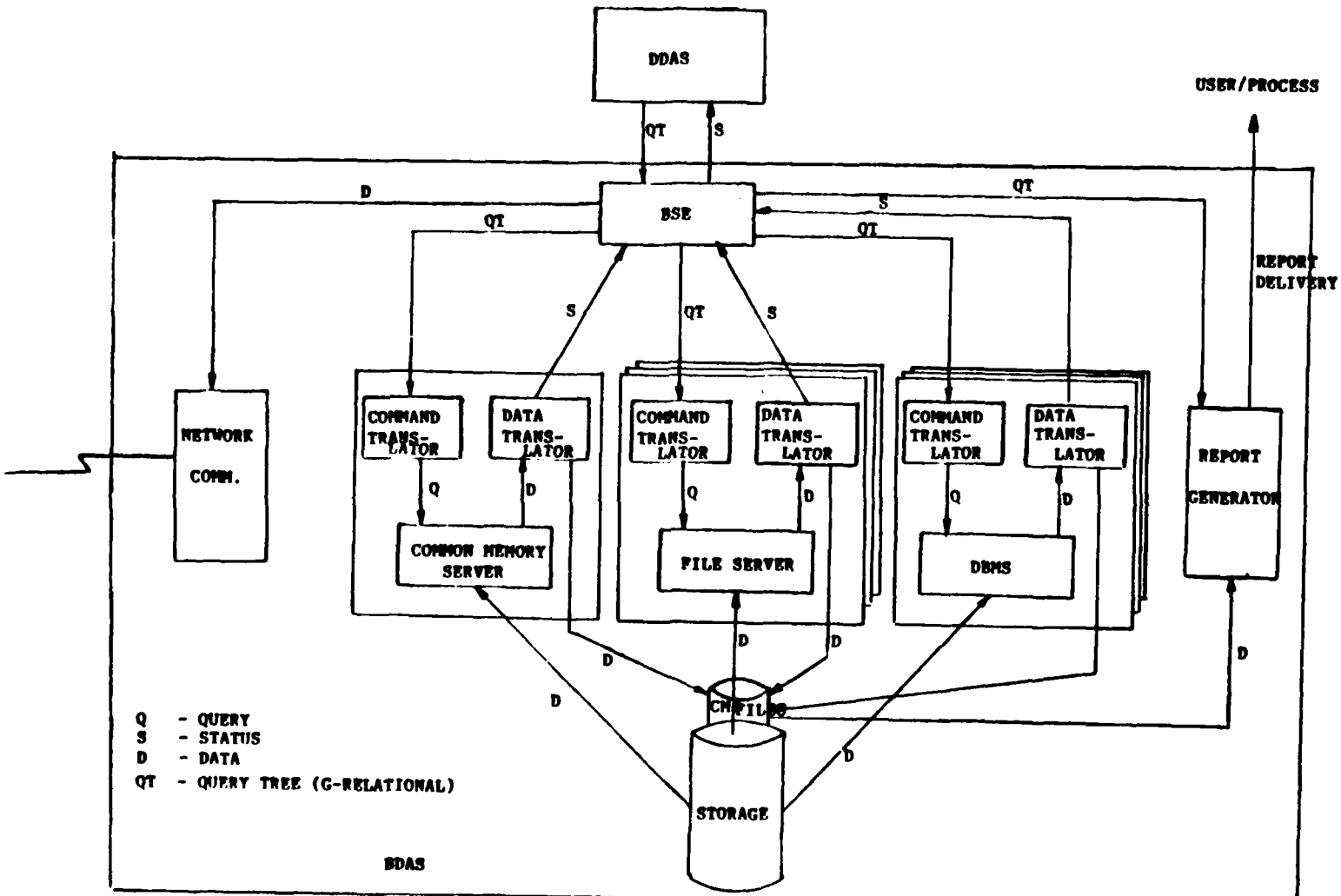


Figure 6. The Basic Data Administration System (BDAS)

d) Data Translation (DT)

Whenever data is moved from one component system to another, it has to be translated from the source representation to the target representation. In a heterogeneous complex, a source-to-common, and then common-to-target conversion of the data is more economical. In IMDAS, the common representation of data is the standard Generalized relation (G-relation). Also, the data exchanged between the various IMDAS modules is in the standard G-relation. The Data Translator (DT) therefore has to do the following tasks: 1) Once the primitive operations are executed by the local data manager or file manager in a BDAS, the retrieved data conforms to the representation used by that local data manager. The DT has to convert this data to its global format, i.e., to the standard G-relation; 2) conversely, when a BDAS receives data to be inserted into its local database, the DT translates the data from its G-relation format to the format used by the local data manager.

e) Report Generator

The requesting process receiving the data may want it to be formatted according to certain format specifications as specified either in the GDML or in the external view definition of that process. Every BDAS has a report generator to perform this operation.

f) Interprocess Communication

If the component system houses multiple control and sensory processes, or if the communications and data management services are implemented as separate processes, a mechanism for communications among these processes is needed. This mechanism can either be process-to-process message passing or some kind of shared memory, depending on the capabilities of the local operating system. The AMRF uses a common memory scheme [LIB85, MIT84], in which an originating process stores information of a particular kind for a particular recipient into a designated area of the common memory, and the process interested in this information retrieves it from this designated area. This scheme is implemented on different systems by either a hardware shared memory, a message passing to a memory manager process, or a process which copies the information between each process's local memory areas and a background common memory [BAR82]. The common memory approach permits data to be used by more than one process without any explicit action by the originator to deliver the data to all consumers. The common memory not only serves as a communication vehicle but also

serves as a database. This approach is effective in equipment level functions which perform real-time operations where the system response time requires much of the data to be memory resident.

g) Network Communication

Every component system must have access to the factory data network and must implement sufficient protocols to enable it to communicate with the other component systems. The IMDAS architecture presumes that the factory data network utilizes a bus or ring local area network technology, thereby providing a peer relationship between component systems. The four lowest layers of the ISO/OSI reference model must be implemented for each component system, either within the system or with network appliances. These basic protocols allow reliable communication between any two systems on the network. In addition, the interprocess communication mechanism chosen within a system must be reflected in the interprocess communication between systems. It can be either direct message passing or a simulated global shared memory.

In the AMRF, the network communication function is performed by a Network Interface Process (NIP) in each component system [MIT84]. The NIP maps the needed areas of a conceptual "global" common memory into the local common memory by replicating areas of its local common memory into the common memory areas of remote components requiring copies. This is done with the assistance of the network and the remote NIP. The NIP is table driven, so that mappings can be dynamically modified by creating or destroying entries in the delivery table. The advantage of this scheme and the local common memory is that they are distributed database mechanisms and therefore can both contribute to and profit from other database techniques in the IMDAS.

5.2. DISTRIBUTED DATA ADMINISTRATION SYSTEM (DDAS)

The DDAS provides uniform access to the integrated database for the control and user processes. In a CIM environment, in each separable group of component systems, i.e. each group capable of independent operation, at least one component system must contain a DDAS. The functions of the DDAS are: data manipulation language service, query mapping service, data assembly, dictionary initialization, and distributed transaction management. Figure 7 illustrates the various DDAS modules and their interrelationships. The functions of each module are described below.

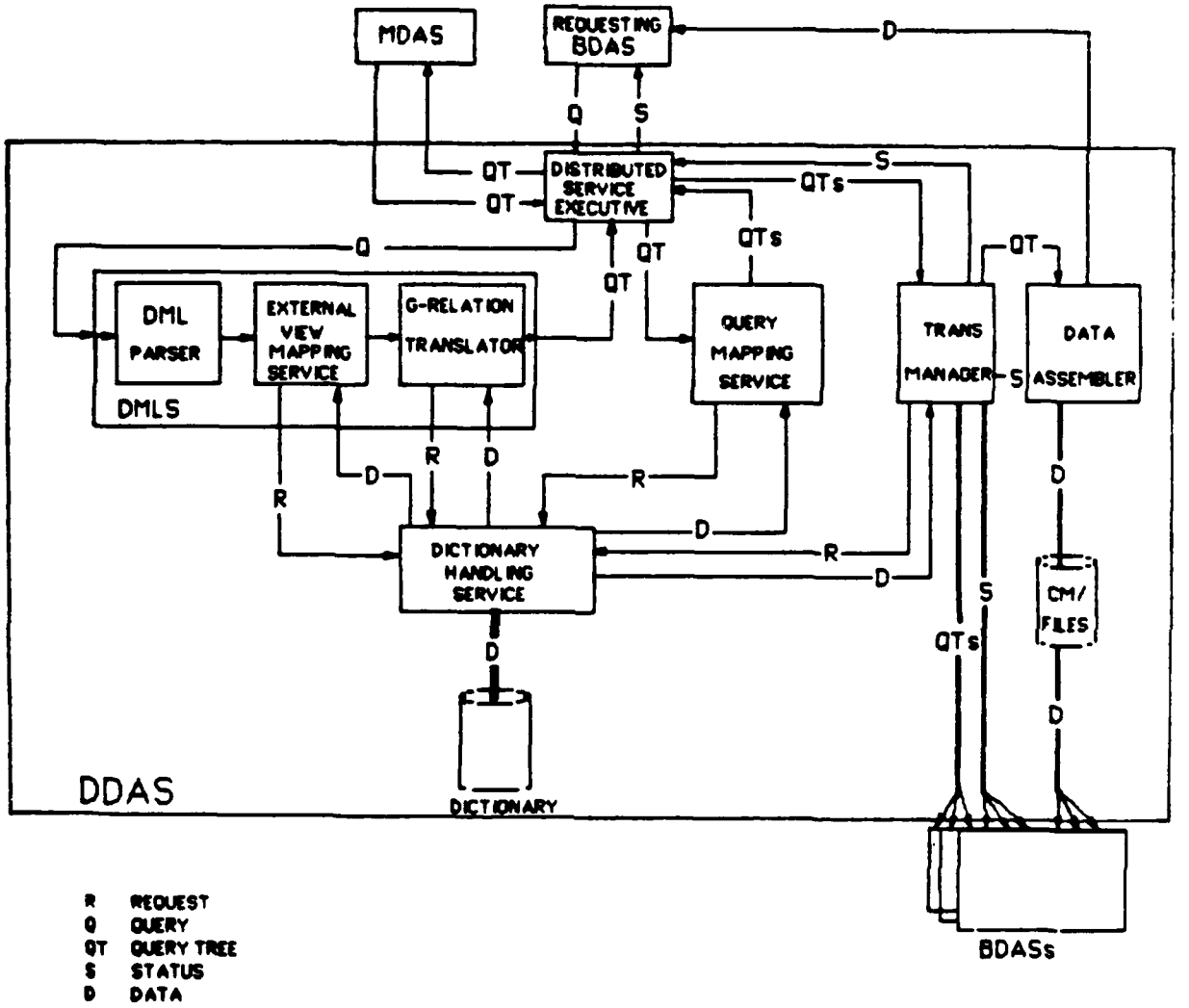


Figure 7. The Distributed Data Administration System (DDAS)

Distributed Service Executive (DSE)

The DSE acts as a single point of contact between the DDAS and the MDAS. The function of the DSE is to co-ordinate the activities of the various DDAS modules. It first initializes its data dictionary to reflect only the data which is resident in the local BDAS. It then creates connections to its subordinate remote BDASs and incorporates their data dictionary information. Any inconsistencies within the dictionary are resolved. Then, on request from the MDAS, a connection to the designated MDAS is established and it transmits dictionary entries containing a model of its local fragment of the global conceptual view. It then executes MDAS directed recovery transactions, if any. Finally, it initializes the interface to control processes served by the DDAS. However, it has to wait for the MDAS to acknowledge a "READY" state before it can begin accepting requests for service.

The DSE then receives the user or process issued GDML through local or networked interprocess communications and forwards it to the Data Manipulation Language Service (DMLS). It then routes the parsed and translated query returned by the DMLS to the Query Mapping Service (QMS). And finally, it passes the list of sub-queries returned by the QMS to the distributed Transaction Manager.

Data Manipulation Language Service (DMLS)

GDML queries are issued against global external views of the integrated database. The DMLS is responsible for composing from the query expressed in GDML, which is a high level language, a query tree of primitive operations. This query tree can either first materialize the external view specified and then perform the operations, or modify the query tree into an equivalent query on the global conceptual view. The query tree is further modified to incorporate integrity and security constraints defined on the underlying data structures. The DMLS then obtains from the DDAS dictionary, information on where the data requested in the query resides. Based on this information, it determines whether the query can be satisfied by the DDAS or if it requires the cooperation of other DDASs. If other DDASs are required, the query tree is forwarded to the MDAS for processing. Otherwise the query tree is passed to the Query Mapping Service (QMS) for further processing.

Query Mapping Service (QMS)

The QMS accepts the modified query tree pertaining to the global conceptual view of the integrated database and performs its mapping to the fragmented views. From the dictionary it accesses the information pertaining to the partitioning and replication of global conceptual data in the local BDASs and modifies the query tree to incorporate the fragments of the data referred therein. Based on the site allocation of the data fragments and the sites' capabilities, it decomposes the query tree into subquery trees, each involving operations on a single BDAS. The decomposition is done in an optimized way so that each subquery consists of an optimum sequence of operations. Each subquery is tagged with its delivery information (i.e. source and destination areas, buffer sizes, and delivery modes) and a list of identifiers of other subquery trees which must be completed before this one. The list of subquery trees generated is directed to the Transaction Manager (TM).

Transaction Manager (TM)

The TM controls and manages the distributed query execution and distributed consistency, integrity, and recovery. When the TM receives the list of subquery trees from the QMS, it initializes the transaction as the atomic unit of recovery and realizes a serializable schedule for the operations. It then analyzes the transaction and checks for any concurrency conflicts with any currently executing transactions. If conflicts are detected, the TM defers the transaction until all involved BDASs finish the transaction which was in conflict. If the transaction was received from the MDAS, the TM must notify the MDAS and wait for the MDAS to direct aborting or stalling the query execution. When necessary, the TM inserts concurrency controls into the transaction to ensure consistency among the BDASs activities and transmits the operation sequences along with directions for constructing delivery paths to the proper BDASs using the local or networked interprocess communication. When the final results of the transaction are correctly received by the requesting process, or when the data is properly updated, the TM identifies the transaction to be complete.

The TM is responsible for the recovery of executing transactions and the integrity of all data being modified. The TM performs transaction and database recovery by selectively delegating recovery actions to the BDASs, properly maintaining distributed checkpoints, and directing parallel activities of the BDASs using the two-phase commit protocol.

Data Assembler (DASM)

The DASM [KHA86] performs selections, joins, assembly, merging, sorting, and other final operations on the data retrieved from the different BDASs. It is required in the DDAS to provide these services for BDASs which have only very basic data management capabilities. It is also required when data from multiple sources are to be assembled into the final result.

Data Dictionary Service (DDS)

The data dictionary in DDAS includes the following metadata: 1) the schema and mapping information for the conceptual-to-fragmented and the fragmented-to-local external views of the data residing on subordinate BDASs, 2) the mapping information for the external-to-conceptual views referenced by the control processes, which this DDAS supports, 3) the security and integrity constraints associated with the data in subordinate BDASs, 4) data delivery information and site information, and 5) the information representing the capability of each subordinate BDAS. When requested by the various software modules of the DDAS, the DDS performs a lookup of the dictionary and provides the necessary information.

5.3. MASTER DATA ADMINISTRATION SYSTEM (MDAS)

There is one MDAS in each independently operating partition of the factory. When the partition is initialized, one of the DDAS is designated as the MDAS and other DDASs are directed to report to it. The function of the MDAS is to coordinate the activities of the multiple DDASs. The coordination consists of managing the master data dictionary, resolving concurrency problems between DDASs, directing initialization, integrity and recovery. The software needed to accomplish these functions is resident in every DDAS. Hence, the MDAS is primarily DDAS software that operates on different subordinate components, i.e. DDASs instead of BDASs, with a master layer dictionary that contains the integrated view of the global database together with the mapping of this global database to its fragments represented by the DDASs.

6. IMDAS PROCESSING SCENARIO

A processing scenario of a typical integrated database access within a DDAS is shown in Figure 8. All retrievals from the database are based on the external views defined for the integrated database. In this

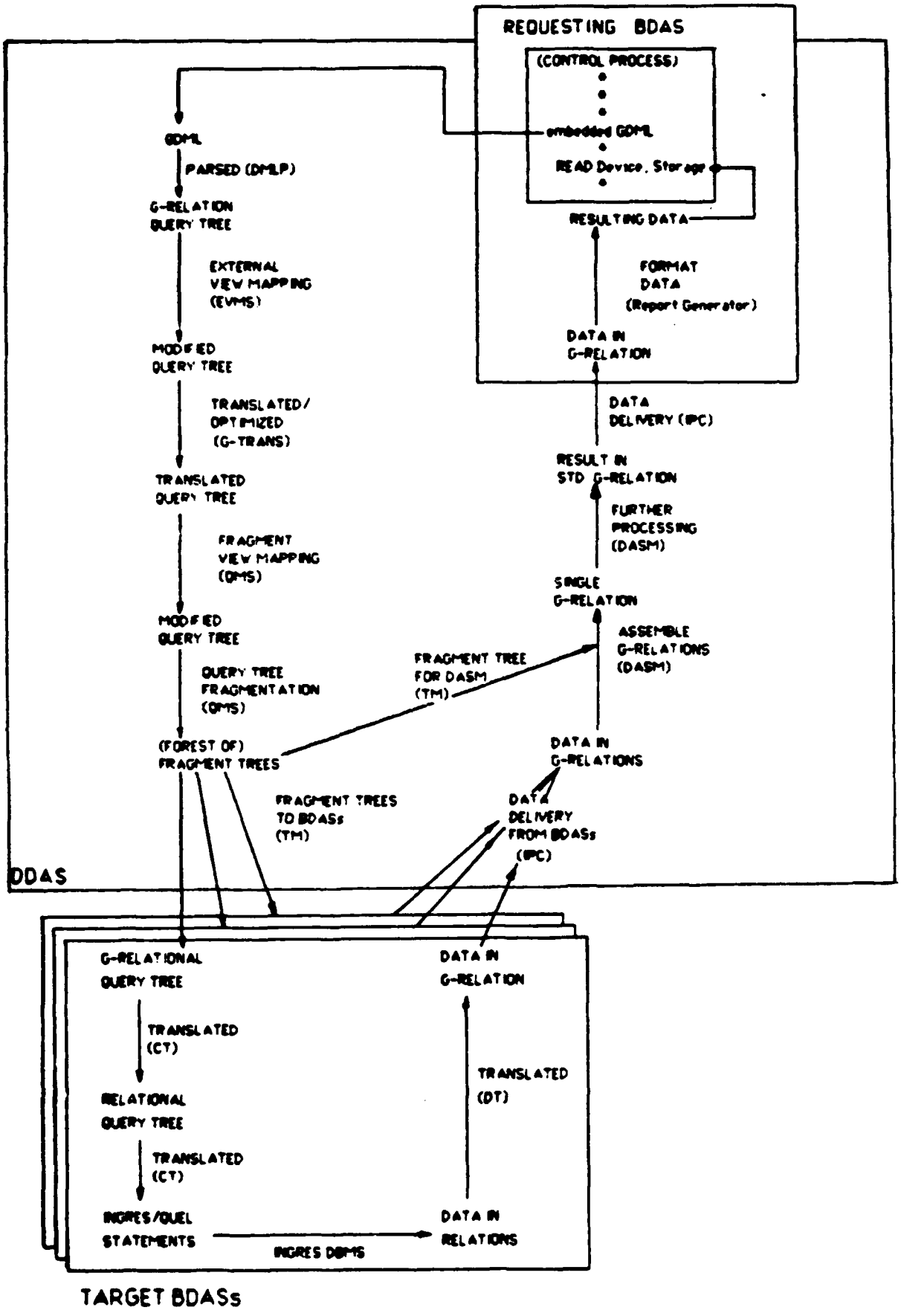


Figure 8. A Typical IMDAS Processing Scenario

illustration, a control process within a component system (represented by requesting BDAS in Figure 8) is requesting an access to some data. The request is in the form of a GDML query. This query is forwarded to the DDAS which controls the requesting BDAS.

Within the DDAS layer, the Data Service Executive (DSE), which acts as a liasion between the various modules of the DDAS, forwards the GDML query to the Data Manipulation Language Service (DMLS). The DMLS first parses the GDML query and generates an equivalent query tree consisting of operations on the internal G-relational representation of the schema. The operators in the query tree are G-relational algebra operators [KRI87]. The query tree is then evaluated against the information present in the data dictionary to determine whether all the data requested is contained in the underlying BDASs; if not, the DSE then routes the query tree to the MDAS. However, once the data is determined to exist in the underlying BDASs, the query tree, which pertains to some external view of the database, is modified with the help of the external view mapping information present in the dictionary to conform to the global conceptual view of the database. The query tree is then further modified, if necessary, to take care of specifying complete paths to the attributes in the query tree. Finally, the query tree is optimized and returned to the DSE. The DSE then forwards the fully grown tree to the Query Mapping Service (QMS).

The main task of the QMS is to handle the fragmentation of the G-relations pertaining to the conceptual schema that is present in the underlying BDASs. The QMS makes use of the fragmentation mapping information present in the dictionary to modify the query tree so that it contains operations on the fragment G-relations. The next task of the QMS is to fragment the query tree itself based on 1) the allocation of the fragmented G-relations to the various underlying BDASs and 2) the operational capabilities of the BDASs involved so that each tree fragment can be sent to the appropriate BDAS for processing. The QMS attaches input or output delivery nodes at the point of "cut-off" of the tree fragment so that the processor, which will execute a tree fragment, will know where to deliver or retrieve the data. The delivery nodes in the query tree essentially carry information as to where and how the data should be delivered or obtained. The process of fragmenting the query tree may result in a tree fragment that has to be sent to the Data Assembler (DASM) either to assemble the data returned from the individual BDASs involved or to perform further operations on them, if necessary. Since there may be a order of precedence involved in the processing of the fragmented query trees, the QMS clearly marks each tree with its precedence number.

including the information about its destination BDAS. This information is essential for the Transaction Manager (TM) in scheduling and dispatching the query trees for execution. The forest of fragment trees thus generated is then returned to the DSE.

The Transaction Manager (TM) obtains the forest of fragmented query trees from the DSE and schedules them for execution. The entire forest of fragmented query trees is handled as a single transaction and the two-phase locking scheme is adopted by the TM to exercise concurrency control. Standard "distributed" recovery and rollback schemes [CER84] are used.

Once the DDAS has finished producing the appropriate query tree for execution at a BDAS, the Basic Service Executive (BSE) of the BDAS, based on the information provided by the DDAS, calls the appropriate Command Translator (CT) to translate the query to the appropriate language that the designated data management system can understand. For example, if INGRES is the data management system, then the query tree of G-relational operations is mapped to an equivalent set of operations on relations [BLU85, KHA86]. The information for mapping between the G-relational schema to the relational schema is stored in the local dictionary. The query tree of relational operations is then translated into QUEL statement(s) which is then executed by the INGRES DBMS. Once the data is returned by the local Data management system, the Data Translator (DT) for that system converts the data format to the standard G-relational format. The data is then delivered to the designated area specified by the delivery node in the query tree. Once this is done, the BSE informs the DDAS Transaction Manager of the completed assignment. If no further processing was required and if only one BDAS was involved in processing the query, the DDAS would have indicated that BDAS to deliver the resulting data directly through inter-process communications to the requesting BDAS. Otherwise, once all the BDASs involved in processing a query return the data, the TM in the DDAS instructs the Data Assembler (DASM) to assemble the data and further process it if necessary. Finally, the resulting data in the standard G-relational form is delivered to the requesting BDAS.

The above description of IMDAS processing is typical for any retrieval from the intergrated database. For certain database updates, however, the scenario changes somewhat. Here, by database updates we mean "insertion", "deletion" and "update" of instances in a G-relation. (In the DML this would be expressed

as insertion, deletion and update of instances in OSAM* class). Database updates that contain a condition for selecting the tuples to be updated are carried out in two phases. Due to horizontal and vertical partitioning of data in the fragment G-relations, the data to be updated may be in one or more fragments and the update conditions may be based on attributes present in different fragments. However, every fragmented G-relation of a conceptual G-relation contains the common instance identifier field. Hence, in the first phase, the DDAS creates a retrieval query tree to select the instance identifiers from the relevant fragments that satisfy the given conditions in the update query. In the second phase, updates are performed on the relevant fragments for those instances whose identifiers were selected in the first phase.

7. CONCLUSION

A prototype of the IMDAS architecture has been developed and integrated with the AMRF testbed facility of the National Bureau of Standards [MCL86]. The description of an earlier prototype can be found in [SU86]. Extension to the SAM* model and the SAM*DML have been made since then to provide better modeling power and expressive capabilities. These changes are being incorporated in a new system under implementation.

Our future research will continue to concentrate on CAD/CAM data modelling and a better interface for design/manufacturing systems. Distributed techniques for query processing, optimization, concurrency control, system initialization, and error recovery, which are critical for integrated factory networks, will also be improved. Knowledge in the form of rules, constraints, and operations is a critical commodity in an integrated manufacturing environment. It needs to be shared by the various components of CIM. Therfore, we are looking into techniques for combining data and knowledge management supports for the CIM. We are also investigating an Object-Oriented approach to the integration of distributed systems.

Acknowledgements

This research is supported by the Navy Manufacturing Technology Program through the National Bureau of Standards grant number 60NANB4D0017. We acknowledge the contributions of the following persons to this project: Mohammad Khatib and Don Libes of National Bureau of Standards; Ashish Kumar and Chong Soo of University of Florida.

REFERENCES

[ALB81] Albus, J.S., Barbera, A.J., Nagel, R.N., "Theory and Practice of Hierarchical Control," Proc. of 23rd IEEE Computer Society International Conference, Sept. 1981, pp. 18-39.

[ANS86] American National Standard Database Language SQL, Document Number X3H2-87-8, ANSI working draft, Dec. 1986.

[APP85] Appleton, D.S., "The Technology of Data Integration", Datamation, pp. 106 - 116, November 1985.

[BAR82] Barbera, A.J., Fitzgerald, M.L., and Albus, J.S., "Concepts for a Real-Time Sensory Interactive Control System Architecture," Proc. of the 14th Sotheastern Symposium on Systems Theory, April 1982.

[BAR86] Barkmeyer, E., Mitchell, M., Mikkilineni, K.P., Su, S.Y.W., and Lam, H., "An Architecture for Distributed Data Management in Computer Integrated Manufacturing," United States Department of Commerce report number NBSIR 86-3312, January 1986.

[BLU85] Blumenthal, D.F., "An Algorithm to Translate Relational Algebra Queries into QUEL", Master's Thesis, Department of Computer and Information Sciences, University of Florida, May 1985.

[BRA82] Bray, O.H., "Distributed Database Management Systems", Lexington Books, 1982.

[CER84] Ceri, S., and Pelagatti, G., "Distributed Databases: Principles and Systems", McGraw Hill Co., 1984.

[CERI84] Ceri, S., Pernici, B., Wiederhold, G.,"An Overview of Research in the Design of Distributed Data Bases", Database Engineering, Vol. 7, No. 4, pp. 46 - 50, 1984

[FUL80] Fullerton, G., "Integrated Program for Aerospace Vehicle Design (IPAD)," NASA. TM-81874. 1980.

[GAD87] Gadient, A. J., "Engineering Information Systems: Implementation Approaches And Issues", Proceedings of the Third International Conference on Data Engineering, pp. 576 - 578, February 1987.

[HAT83] Hatvary, J., Merchant, M.E., Rathmill, K., and Yoshikawa, World Survey of CAM, Butterworth & Co. Ltd., Kent, U.K., 1983.

[ICA79] ICAM Program Prospectus, Air Force Systems Command, Wright-Patterson Air Force Base, OH, USA, September 1979.

[ICA83] ICAM Project, "Computer Program Development Specification (DS) for ICAM Integrated Support System (IISS) Configuration Item: Precompiler," prepared by Control Data Corporation and D. Appleton Company, Dec. 1983.

[KHA86] Khatib, Mohammad, "Data Assembly in a Distributed Heterogeneous Data Base Management System", Master's Thesis, Electrical Engineering Dept., University of Florida, May 1986.

[KRI85] Krishnamurthy, P., "A Data Manipulation Language for the Semantic Association Model SAM*." Master's Thesis, Electrical Engineering Department, University of Florida, Dec. 1985.

[KRI87] Krishnamurthy, V., Internal Report, Database Systems Research and Development Center, University of Florida, April 1987.

[LIB85] Libes, D., "User-level Shared Variable", Proc. of Summer 1985 USENIX, Portland, Oregon, June 1985.

[MIT84] Mitchell, M. and Barkmeyer, E., "Data Distribution in the NBS Automated Manufacturing Research Facility", Proceedings of the National Symposium on Advances in Distributed Data Base Management for CAD/CAM, NASA Publication 2301, April 1984.

[MCL83] McLean, C.R., Mitchell, M., and Barkmeyer, E., "A Distributed Computing for Small Batch Manufacturing Sysytems," IEEE Spectrum, May 1983.

[NAN84] Nanzetta, P., "Update: NBS Research Facility Addresses Problems In Setups For Small Batch Manufacturing", Industrial Engineering, pp. 68-73, June 1984.

[NEU84] Neuhold, E. J., "Distributed Database Systems with Special Emphasis Toward POREL", IPAD II.

Proc. of a National Symposium, NASA Conference Publication, 2301.

[SIM82] Simpson, J.A., Hocken, R.J., and Albus, J.S., "The Automated Manufacturing Research Facility of the National Bureau of Standards," Journal of Manufacturing Systems, Vol. 1, No. 1, 1982.

[SMI86] Smith, B., "PDES Project: Objectives, Plans and Schedules", Internal Memorandum on Product Data Exchange Specification, National Bureau of Standards, June 1986.

[SU83] Su, S.Y.W., "SAM*: A Semantic Association Model for Corporate and Scientific-Statistical Databases," Information Sciences, 29, 1983, pp. 151-199.

[SU85] Su, S.Y.W., "Modeling Integrated Manufacturing Data Using SAM*," Proc. of Data Base Systems for Office, Engineering and Science, Karlsruhe, West Germany, March 1985 and IEEE Computer, Jan. 1986.

[SU86] Su, S.Y.W., Lam, H., Khatib, M., Krishnamurthy, V., Kumar, A., Malik, S., Mitchell, M., Barkmeyer, E., "The Architecture And Prototype Implementation of an Integrated Manufacturing Database Administration System", Spring COMPCON 1986

[SU87] Su, S.Y.W., Krishnamurthy, V., Lam, H., "An Object-Oriented Semantic Association Model (OSAM*) for Engineering and Manufacturing Databases", in preparation.

[WED87] Wedekind, H., Zoerntlein, G., "Conceptual Basis for Database Application in Flexible Manufacturing Systems (FMS)", Proc. of IEEE International Conference on Robotics and Automation. Vol. 1, pp. 551 - 557, March 1987.

[WIL82] Williams, T., et. al., "R*: An Overview of the Architecture, Proc. of the International Conference on Database Systems", Jerusalem, June 1982.

[YOS81] Yoshikawa, H., Rathmill, K., and Hatvany, J., "Computer-Aided Manufacturing: An International Comparison," National Academy Press, Washington, D.C., 1981.

REFERENCE MODELS FOR INFORMATION ENGINEERING

Paul Thompson

Control Data Corporation

Contributed by Paul Thompson

1.0 INTRODUCTION, PURPOSE AND OVERVIEW.

1.1 Introduction and Purpose.

This paper describes three important Reference Models and one Engineering Framework for data sharing and exchange. It relates Reference Models to standards activities and shows how to integrate them. This integration is not widely recognized.

It also gives examples of technical models to illustrate the content of an Engineering Framework approach. The context is engineering and manufacturing enterprises.

1.2 Overview.

First we present an overview of the current state of industrial automation and the need for information engineering techniques. Then we discuss three Reference Models for standards activities: The ISO/Open Systems Interconnection, the ISO/Conceptual Schema Concepts and the ANSI/Three Schema Architecture. Next we present the PRECISE * Engineering Framework for data-driven conceptualization, specification, design and development. We illustrate this framework with two Technical Models. Finally we describe directions for future work.

2.0 ASSESSMENT OF THE CURRENT SITUATION.

2.1 Historical Trends and Pressures.

2.1.1 Economic Developments.

Today's engineering and manufacturing enterprises face new economic developments.

We see increasing pressure from competitors, unstable interest and foreign exchange rates, speed of technology transfer, low offshore labor costs, new materials and processes, changing markets, new environmental laws and so on.

The pace of economic change shows clearly in the record numbers of new business start-ups, mergers and bankruptcies of the last few years.

2.1.2 Information System Developments.

2.1.2.1 Industrial Automation.

Two important technological changes for the engineering and manufacturing community are the dramatic decrease in cost of and the enormous increase in capability of information systems hardware and software. These two forces drive the change from an industrial age to a mixed industrial/information age.

As a result, information systems hardware and software are becoming very important to managers, engineers and factory workers of all kinds.

First, we are designing more intelligence into the end product of the enterprise. For example, digital dashboards, voice warnings and computer controlled ignition and diagnosis in new automobiles. This intelligence provides added value to the product.

Second, we're putting more automation into the design and manufacturing processes for better quality, and more flexible and more efficient production. Examples are assembly robots, flexible manufacturing cells, engineering design analysis programs, expert system diagnosis, product and configuration databases, CAD/CAM/CAE support tools and so forth.

Third, we're transmitting more data electronically between the enterprise, its divisions, suppliers and customers. We're wiring together the "islands of automation" inside and outside the enterprise. Even firms which have believed automation was too expensive for them are now being dragged into the information age as their important customers insist upon electronic data exchange.

2.1.2.2 Computers and Information Systems Science: Yesterday and Today.

In the early days we spent our time on algorithms to crunch numbers and on building data processing systems to track the accounting figures. We located data processing systems with the administrative offices, away from the day to day activities of the enterprise. We considered computers a separate department. We didn't consider them to be part of our business, engineering and manufacturing processes.

Today we're concerned with databases, expert systems, graphics, communications and on-line integrated systems. Engineering and manufacturing divisions, not the administrative offices, are now taking the lead in developing and installing advanced computer applications. We're bringing advanced automation directly to the workers.

In the past it was sufficient for a programmer to learn a scientific or business progra ng language to do useful work. Today the computer specialist has to be an expert in many information system technologies and modeling techniques.

2.2 Need for Advanced Information Engineering.

However, despite the power of today's computer hardware, we continue to have problems developing applications that meet the users' and organization's needs. Information systems construction remains a complex, labor intensive and high risk operation.

Some reasons for development problems include:

- Good work demands a thorough understanding of both the users' world and the computer world. (One barely has time to be an expert in even one of these areas).

- Rapid technological advances quickly outdate our current knowledge.

- Vendors promote their own unique systems, ignoring or diluting standards.

- The technology of computers is so fascinating that many of our best people become experts *in the technology before*, and instead of, understanding the problems they have to solve.

- The enormous labor required to build and debug the systems of the past acts as a brake on the introduction of new work methods and development solutions. (We call this the legacy problem.)

Of all development problems, one problem stands out: the difficulty of truly integrating systems so the various parts work together intelligently and synergistically.

It isn't hard to buy stand-alone applications. It isn't hard to wire the hardware together. But it is really hard to get the systems to communicate intelligently with each other. And it's really hard to build them sound enough that they can grow and change with the enterprise without rewrites.

Information Engineering is that branch of information systems science that offers some solutions to these problems. Information Engineering is an approach to building integrated systems based on Reference Models and Modeling Methodologies. Information Engineering builds sound, truly integrated systems.

Information Engineering develops sound, integrated information systems the same way our best companies build their own complex products: by using an engineering process to determine objectives

and requirements, to specify and analyze designs, to prototype products before production, and to manage the process by objectives, costs and schedules.

Like other forms of engineering, Information Engineering uses modeling and analysis technologies appropriate to the development phase. These modeling and analysis technologies come from basic information science fundamentals.

Reference Models capture and delineate these principles for ease of learning, discussion and standards making.

3.0 INFORMATION ENGINEERING REFERENCE MODELS.

A Reference Model prescribes the important, fundamental form of a subject area. It identifies major concepts, functions, purposes and building blocks. A Reference Model sets direction for future project and standards work activities. These work activities may develop one or more standards under the Reference Model.

Here we discuss three categories of Reference Models used in Information Engineering. These Standards Reference Models originate from national and international standards activities. Most deal with data organization or communication. Engineering Framework Models identify needed Technical Models and the organization of people and activities necessary to accomplish effective work efforts. The Technical Reference Models specify analysis and design techniques to analyze problems and specify solutions. Most Technical Models have a strong graphical component.

We discuss below a few of the more important examples of each kind.

Major Standards Organizations Models.

3.1 The ISO Open Systems Interconnection Model. [1] [2] [3]

The Open Systems Interconnection Reference Model is a result of work by the International Standards Organization (ISO) and the Consultative Committee on International Telegraphy and Telephony (CCITT). Its purpose is to bring order into the connection of distributed computer systems and the exchange of messages among them.

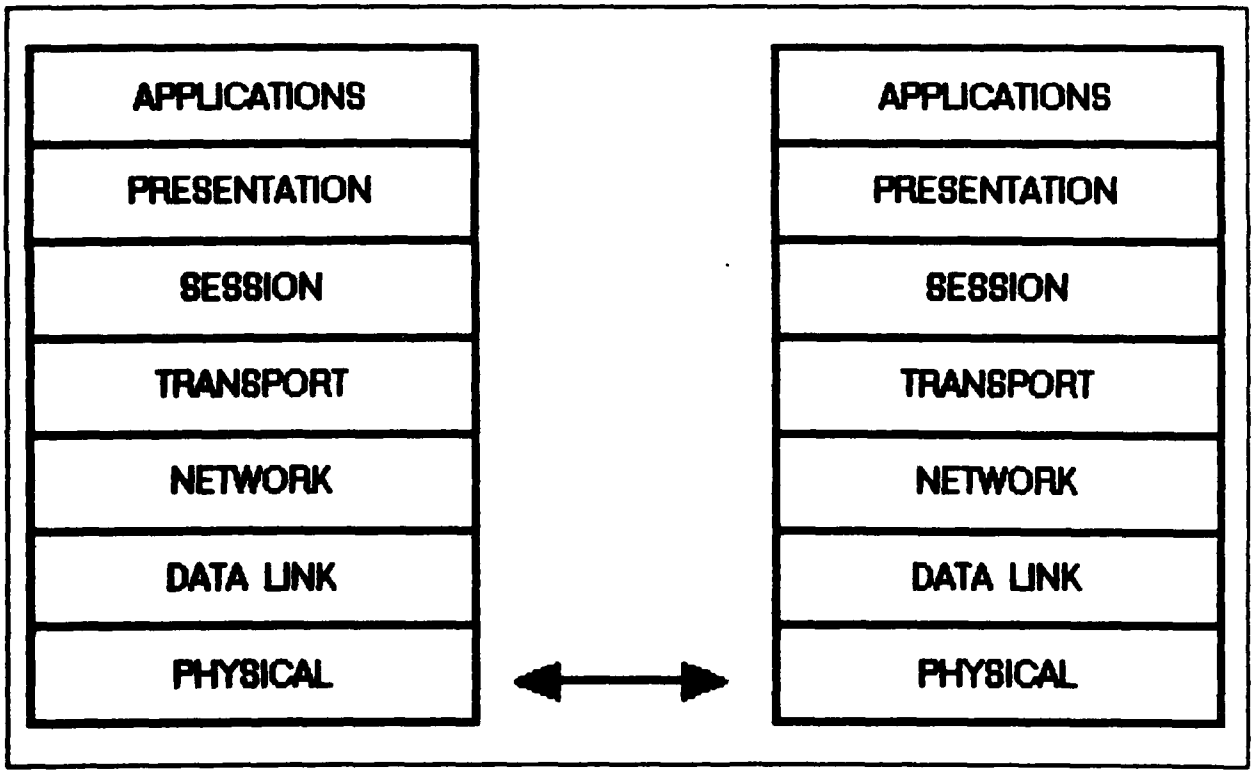
ANSI	American National Standards Institute
BRM	Binary Relationship Model
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CCITT	Consultative Committee on International Telegraphy and Telephony
CDC	Control Data Corporation
CS	Conceptual Schema
DBMS	Database Management System
EAR	Entity Attribute Relationship
ES	External Schema
FFM	Functional Flow Model
FM	Functional Model
IA	Information Analysis
IE	Information Engineering
IEEE	Institute of Electrical and Electronic Engineers
IPL	Interpreted Predicate Logic
IS	Internal Schema
ISO	International Standards Organization
LAN	Local Area Network
NIAM	Nijssen's Information Analysis Method
OSI	Open Systems Interconnection
SPARC	Study Planning and Requirements Committee
SQL	Structured Query Language
TC97/SC5/WG3	Technical Committee-Subcommittee-Working Group

Table 1. ABBREVIATIONS

The ISO/OSI model partitions the communications process into seven layers to provide modularity and ease standardization. Lower layer functions deal with physical communications and links. The higher layer functions deal with functionality and protocols.

More people are probably aware of this model than the other two models. When an organization reaches the point that it wants to tie together distributed computers (often from different vendors) it needs a communication model. This model permits it to evaluate vendor offerings. It gives some assurance that your investment in communication hardware and software won't be wasted.

3.1.1.1 The ISO/OSI Model.



(Figure 1. The OSI 7 Layer Model.)

The seven layers are:

1. physical	- electrical and physical connections network topology.
2. data link	- format of discrete message frames control of error between 2 nodes layer 2 guarantees correct message transmission.
3. network	- routing of packets between networks of multi-link paths flow regulation and status messages.
4. transport	- reliable transparent multi-link paths physical addresses layer 4 guarantees correct packet transmission.
5. session	- communication between user programs device names instead of addresses.
6. presentation	- data restructuring; encryption how applications enter the network.
7. applications	- services for application programs for example, electronic mail.

3.1.1.2 IEEE 802 Standards.

Experts have found it easy to agree upon standards for the lower layers (1-3) of this architecture. Indeed, the Institute of Electrical and Electronic Engineers (IEEE) LAN-802 Standards Committee has proposed three different standards for these layers.

1. 802.3 baseband or broadband transmission over a bus architecture using carrier-sense multiple-access with collision detection. This is the basis of the Xerox Ethernet (tm) and the IBM PC Network.

2. 802.4 token-passing bus architecture.

3. 802.5 token-passing ring architecture, such as the IBM Token-Ring Network.

3.2 The ANSI/SPARC 3-Schema Architecture. [4] [5]

In 1977 the Study Planning and Requirements committee (SPARC) of the American National Standards Institute (ANSI) published a report on requirements for database systems. This report has since become known as the three schema architecture report. In the database world this report has enormous (and still growing) influence.

Previously, all database management proposals defined the properties of data in one database schema with some permitted subsetting into subschemas. These proposals were limited and not powerful enough to handle engineering and manufacturing enterprise problems nor to achieve data independence.

The ANSI/SPARC committee recognized the need for three schema types to express three different views of the data. Today most people agree that a three schema architecture is absolutely essential to integrated information systems.

The three schemas are:

3.2.1 One Conceptual Schema.

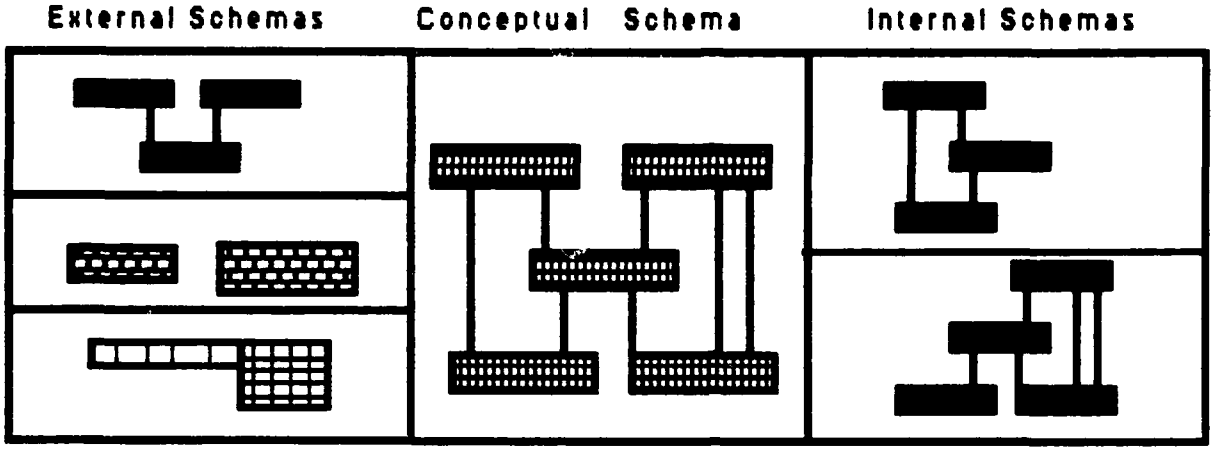
A Conceptual Schema models the enterprise view of the data. The conceptual comprises the WHAT of the enterprise data. The Conceptual Schema must carry the semantics of the data to permit intelligent database systems.

3.2.2 Multiple Internal Schemas.

An Internal Schema describes the physical structure of the data on one or more computers, using various database structuring packages. Internal Schemas comprise the HOW of efficient physical data storage. With an Internal Schema you can tune an application AFTER it is running. You change only the Internal Schema, not program code. This provides run-time efficiency and development productivity.

3.2.3 Multiple External Schemas.

An External Schema describes the user's view of the data (or his program's view of the data). External Schemas comprise the HOW of data usage. Different groups within the enterprise want to view the data differently. They can when they use External Schemas. Also some applications run better with a relational table view of data and some with a record-at-a-time network view of data. A programmer can design the best external schema for his application.



(Figure 2. The ANSI/SPARC 3 Schema Architecture.)

3.2.4 Relationship of ANSI/SPARC 3-Schema Architecture to Database Standards.

At present the only database language standard is SQL. SQL is both an ANSI and ISO standard. The CREATE TABLE statements of SQL correspond to an ANSI Conceptual Schema, although a very simple one and a semantically weak one. Other SQL statements create tabular VIEWS. These are somewhat equivalent to the ANSI External Schema. SQL has no Internal Schema.

3.3 The ISO Conceptual Schema Concepts. [6] [7]

The most important ANSI/SPARC schema is the Conceptual Schema. The Conceptual Schema constitutes both a model of the enterprise and an information contract between the modelers and the implementors of a database system. To be effective you need a powerful Conceptual Schema. With it you can build intelligent database systems. You can reduce development costs. You can provide a stable platform for you applications.

It is also important to have an equally powerful Information Analysis method for the design of the Conceptual Schema. More on this later.

ISO committee TC97/SC5/WG3 studied Conceptual Schema needs. They derived a set of concepts and principles for the Conceptual Schema and a set of assessment guidelines for Conceptual Schema

proposals. The committee also considered and evaluated three candidates for the Conceptual Schema language.

Working on a common case study, they documented the principles of the three candidates and evaluated their power.

The table below summarizes the results of their study and evaluation.

	EAR	BRM	IPL
Propositions modeled	25	40	47
Based on	Data	Semantics	Math
Graphical notation	X	X	
Tools	X	X	

(Figure 3. Comparison of Conceptual Schema Models.)

3.3.1 The Interpreted Predicate Logic Model - IPL.

IPL was the most expressive of the three models compared. Based on predicate logic and mathematical notation it was able to describe the entire universe of discourse of the sample problem.

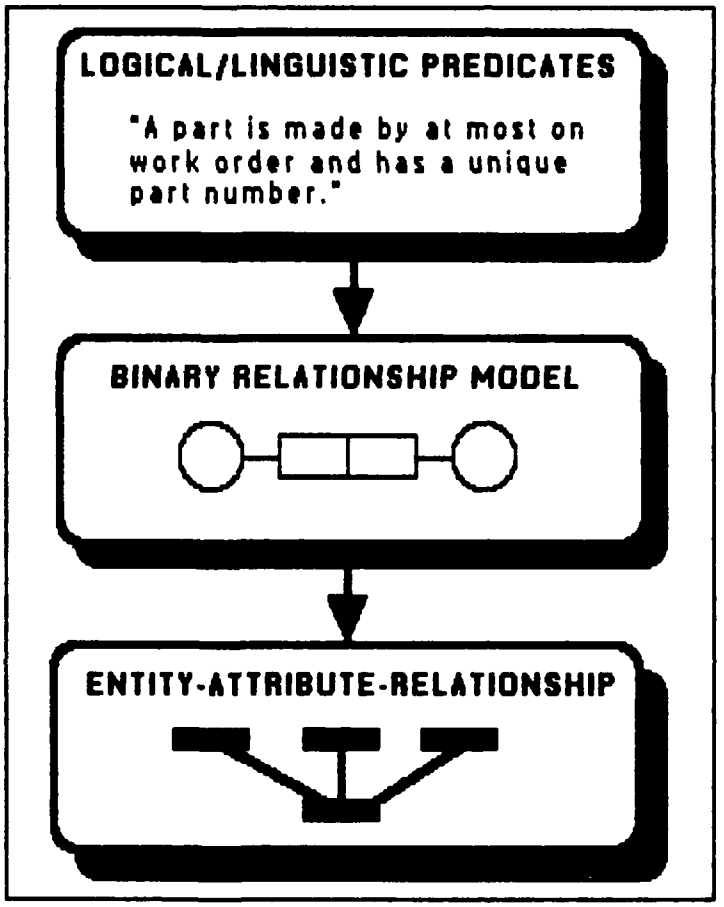
However, IPL has some serious limitations. It doesn't have a graphical notation, it does not have software tool support, and it doesn't come with an Information Analysis method. The degree of mathematization makes the model attractive to mathematicians but acts as a barrier to less highly trained practitioners.

3.3.2 The Binary Relationship Model - BRM.

This modeling technique is more semantically expressive than Interpreted Predicate Logic. It comes with a rich and interesting graphical notation, a strong information analysis methodology and supporting software tools for aid in analysis and database engineering. The model has basic but powerful and graphical constraint rules based on set theory. It also has an advanced constraint language (not documented in the report) which is as powerful as IPL.

Supporting software can automatically categorize entities, analyze

references, migrate keys, fully normalize relations, and generate database schemas.



(Figure 4. The ISO Conceptual Schema Concepts.)

3.3.3 The Entity Attribute Relationship Approach - EAR.

This modeling technique was the least expressive of the three compared although it is the technique most widely used today. Graphical notations and supporting software tools exist for this technique. Somewhat weaker information analysis methods also exist. Interestingly enough, this model is the closest of the three to current database management systems.

Supporting software can provide key migration and transliterate the model to data structures.

3.4 Integration of the Reference Models.

An obvious question is how do the three Reference Models relate? For standards work does not exist in isolation. Project and information engineers have to use their results in a consistent step-by-step approach to development and deployment. Isolated and unconnected Reference Models would be of little practical use.

3.4.1 Integration of OSI and 3-Schema Models.

First let's take the OSI and 3-Schema models.

1. The OSI model is an architectural model for DATA EXCHANGE. The 3-Schema model is an architecture model for DATA SHARING. Data which are exchanged are data in motion. Data which are shared are data at rest. At a certain level of abstraction the user is unconcerned whether the data he receives originate in a shared database or are exchanged (transmitted) from a remote user. He is concerned with the message, not the medium.

2. The OSI model describes HOW the data is physically transported.

The 3-Schema model describes:

- HOW users and programs view the data. (External Schema).
- WHAT the data means. (Conceptual Schema).
- HOW the computer stores the data. (Internal Schema).

The OSI model and the Internal Schema both describe the physical format and access of data.

(The OSI model describes the format in great detail because it aims for open systems. The Internal Schema is just the top structure of a physical data structuring. The lower levels are private the vendors' database management systems.)

So it's clear that the integration point for the two models is the internal schema. In a data sharing environment, the user will store the data in a database. He (or his friendly database engineer) will describe that database with an Internal Schema. In a data exchange environment the user (or his friendly communications expert) will conduct the communication through the OSI 7 layer model.

Looked at from another point of view, you may consider the

External and Conceptual Schemas layers 8 and 9 in the current Open Systems Architecture.

3.4.2 Integration of the ISO/Conceptual Schema Work.

Now let's take integrate the ISO/Conceptual Schema work.

Where the previous two reference models prescribed computer hardware and software architectures, the ISO/Conceptual Schemas is more a development approach, or at least we can view it as such. We can use the results of the ISO/CS report to put together an Information Analysis approach and link the resulting Conceptual Schema to the ANSI/SPARC Conceptual Schema.

The purpose of an good Information Analysis is to develop a rigorously correct, consistent, understood and agreed-upon problem-oriented Conceptual Schema. This Conceptual Schema is both a formal description of the user's information problem and a prescription for any database design.

A problem-oriented Conceptual Schema resulting from an Information Analysis determines the construction of the database Conceptual Schema. That is, if you need database technology. The same problem-oriented Conceptual Schema can just as easily determine a frames structure or an object-oriented database design, when you need knowledge base technology. It can even help you decide between the two! (But that's another story.)

One well-tested way of conducting an Information Analysis is to collect user statements of the meaning of data in their environments, integrate these statements into a semantic network so all relationships are clear, and then to carefully construct a data model from the resulting knowledge. Although it is unlikely that the ISO committee had this method in mind when it examined the three candidate models, the models do lend themselves to this approach.

We can construct an Information Analysis method using the ISO candidate models:

1. Collect user statements or predicates. Analyze these user predicates for fact and rule content. The Interpreted Predicate Model provides a formal mathematical symbolism for this analysis.

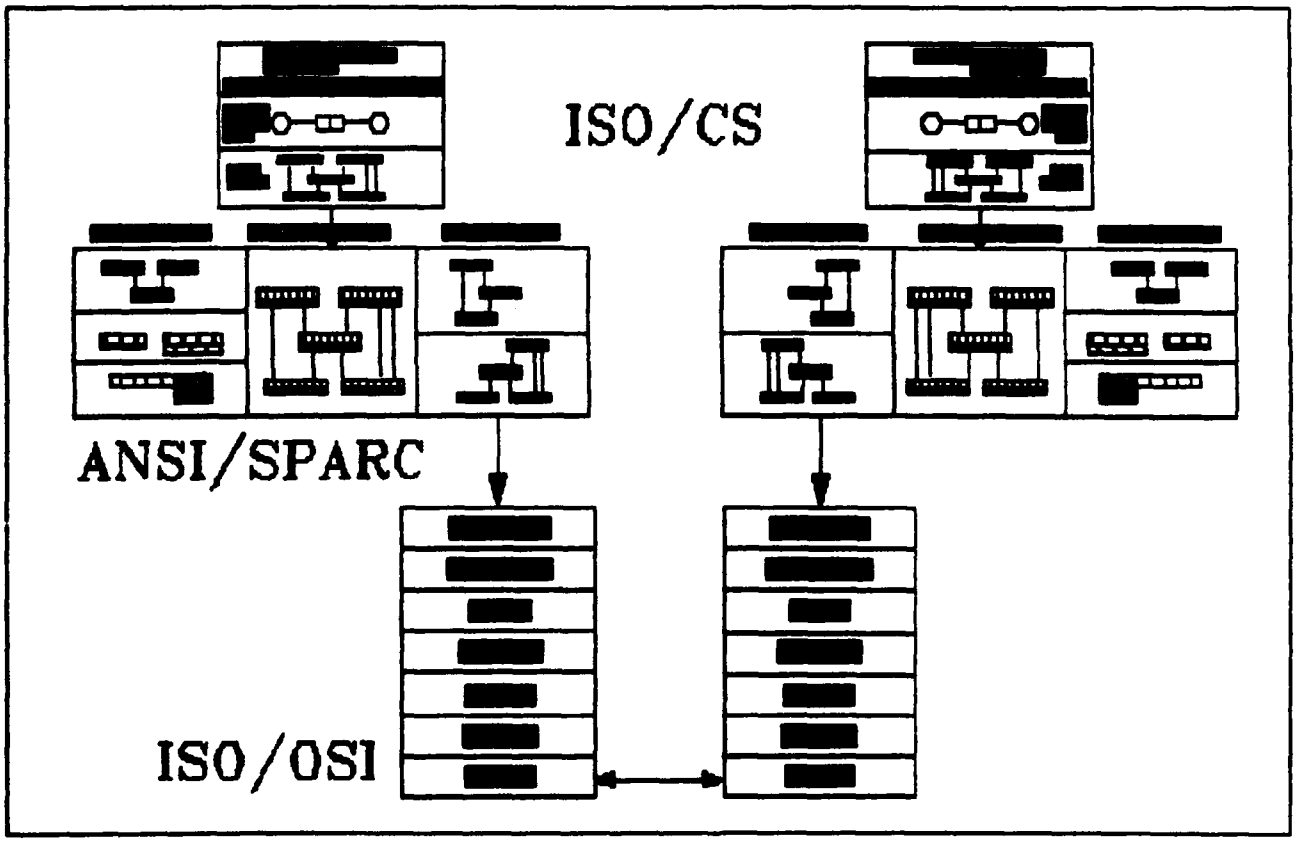
2. Integrate these statements into a semantic network to show relationships and to check for consistency. The binary relationship model provides formal graphical and lexical symbolism for this work.

3. Transform the binary network into a data model. The Entity

Attribute Relationship data model provides a good mechanism for recording data design decisions.

4. If you are going to share data, then transform the data model into a database Conceptual Schema. Next determine the External Schemas and Internal Schemas based on access needs.

5. Or, if you're going to exchange data, determine the products and standards to be used according to the OSI model. Then determine a neutral Conceptual Schema and any External Schemas needed.



(Figure 5. Integration of the Three Reference Models.)

3.5 FRAMEWORK MODELS. [10]

A framework describes in general terms the steps involved in Information Engineering and identifies the modeling techniques or classes of techniques to be used. A framework is a model of models.

There are very few consistently thought-out framework approaches published today. As far as we know, Control Data is the only company that has documented such an approach. We will describe their approach here.

CDC calls their product PRECISE * Information Engineering (tm). The PRECISE approach to engineering Information Systems consists of a framework, a project guide model, methodologies, techniques, tools, training and consulting services.

The PRECISE * Framework satisfies many of the requirements placed on an Information Engineering approach. This framework covers all steps from the strategic planning down to detailed design. It balances input from strategic planners, domain experts and technical engineers. It relates activity analysis and design to data analysis and design. It combines modeling techniques at the conceptual level with data-driven design and implementation.

The outline of the framework are simple but important for its balance. There are three levels: STRATEGIC, TACTICAL and TECHNICAL. Within each level are four models. The upper two models document the WHAT and the lower two models document the HOW. The right two models document the DATA and the left two models document the ACTIVITY.

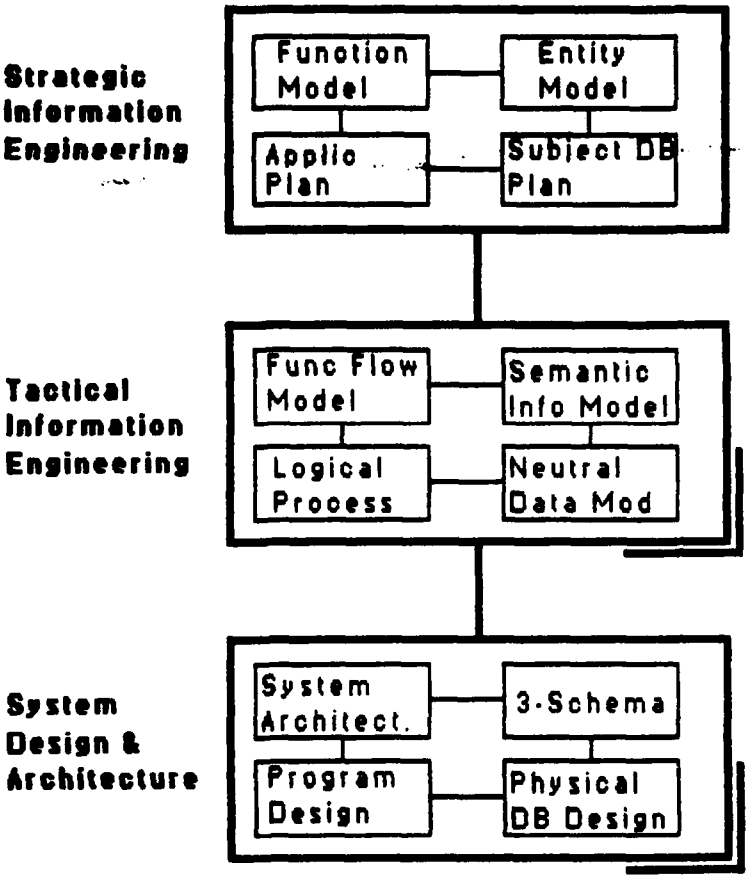
3.5.1 Strategic Information Engineering.

Strategic Information Engineering models the enterprise. It determines its information needs and requirements for information systems.

3.5.1.1 The Enterprise Model.

The Function Model is a hierarchy of the enterprise functions and their information needs. It is independent of current organizational structure, current methods, and current technology. The function model forms a stable foundation for the specification and development of Information Systems capable of surviving both environmental and technological changes.

The Entity Model defines the objects of major interest to the enterprise and the relationships between objects and functions.



(Figure 6. PRECISE * Information Engineering.)

3.5.1.2 The Planning Model.

Since the development of an integrated Information System is a major effort, PRECISE * determines separate and manageable project efforts based on results of the modeling, dependencies, immediate critical needs, and resource availability. It puts together these in the Application Plan and the Subject Database Plan.

3.5.2 Tactical Information Engineering.

Tactical Information Engineering develops thorough and accurate specifications of all user requirements needed to meet strategic and tactical objectives. This phase repeats for each project effort, until the set of integrated requirements specifications is complete.

Tactical Information Engineering distinguishes between a domain and the information system within the domain.

3.5.2.1 Domain or Object System Modeling.

The Function Flow Model is a refinement of the Function Model developed during the strategic planning activity. For the functions that fall within the scope of a particular project, PRECISE * determines and documents the subfunctions, environments and flows.

The Semantic Information Model is a rich semantic network that, as the name implies, concentrates on the meaning of information rather than on data structure. It contains object types, fact types and all-important, protective constraints. PRECISE * uses a unique methodology. It begins with everyday natural language of the domain experts and steps through an exact linguistic analysis until it reaches the semantic network.

3.5.2.2 Information System Modeling.

The Logical Process Model defines the scope of the Information System and specifies the logic of processes to support functions. Functions decompose into either manual processes or automated processes. The model includes information stores, frequency of occurrence, response time requirements, triggers, and authorization.

PRECISE * generates the Neutral Data Model from the Semantic Information Model by strict rules that preserve the model integrity independent of any database structure or product! The Neutral Data Model provides a richness of data structure equal to or beyond any database management product now available. The thoroughness of the data structure enables one to select the most suitable database manager.

To further enhance the value of models, PRECISE * prescribes a Validation Prototyping to confirm the models.

3.5.3 System and Data Architecture.

The System and Data Architecture phase translates specifications into a design. It builds more refined and detailed models based on the previous work. This phase repeats as necessary to yield implementable software projects.

System and Data Architecture distinguishes between the modeling of the technical solution (technology dependent) and the physical solution (product dependent).

3.5.3.1 The Technical Solution.

PRECISE * organizes the specifications of subsystems, procedures, programs, subprograms, etc. into the System Architecture model. Guided by the Function Model, Logical Processes group to form run units. Existing systems map into the design and interfaces connect the old and new. With the help of users, design engineers design screens and reports.

Creating the System Architecture also involves evaluating technical alternatives and selecting distributions of system components and local and remote communications means.

On the data side, the 3-Schema Model separates the enterprise definition of the database from physical storage characteristics and usage views of the data.

The External Schemas derive their data and navigational needs from the logical processes. In turn, the Internal Schemas derive their data needs from distribution analysis and their access mechanisms from the need to support External Schemas joins and navigation and also from Conceptual Schema constraint checking.

A Design Prototype verifies that user interfaces (screen and report layouts) are not only acceptable, but will increase the productivity of the end user.

When needed, the System Design and Architecture phase investigates an application's potential performance with a performance prototype or benchmark.

3.5.3.2 The Physical Solution.

The Detailed Program Design and the Physical Data Model develop detailed, product specific designs. These models specify in detail how to use selected products.

Detailed program design goes down to a pseudo-code level using data-driven design techniques.

The physical data model describes how to store the data. It describes the files, keys, indices, sequences, physical file placement, etc.

3.6 TECHNICAL MODELS. [11]

To illustrate how technical models fit into an Information Engineering Framework, we select and describe two models from the Tactical Information Engineering phase. These are newer and more effective conceptual models than you may be acquainted with. (However, this introduction is so brief we can't even attempt a justification of that statement.)

For each model we describe its purpose, the concepts modeled and the process you use to create it. (We hope to give you an idea of how the modeling works in real life).

3.6.1 Functional Flow Model.

3.6.1.1 The Purpose of a Functional Flow Model.

A functional flow model represents the engineering, manufacturing and management functions essential to the enterprise and the flows between those functions. This model is independent of any specific physical computer system implementation and any specific organizational structure.

With a functional flow model you can quickly grasp the scope and important interrelationships of a domain.

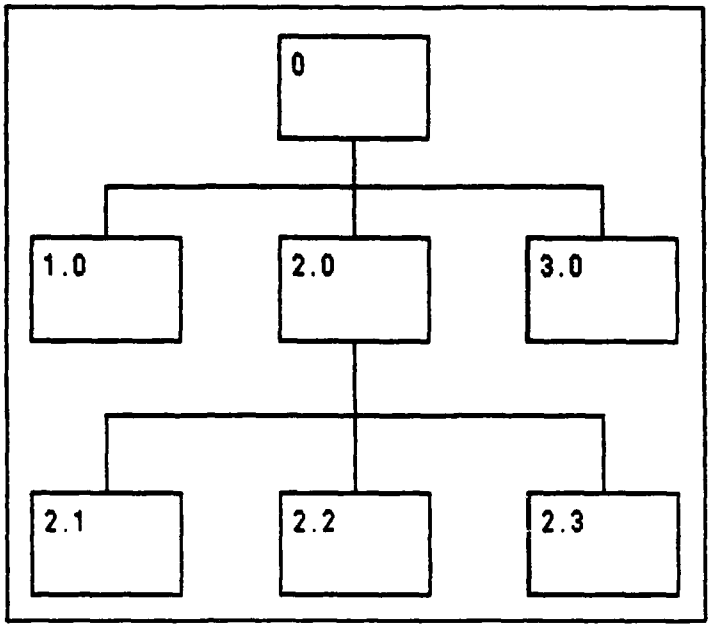
3.6.1.2 Functional Flow Model Concepts.

Function Box - a function of the enterprise. We make no distinction between automated and manual activities, because what you find automated today might not be fitting for tomorrow and what is manual today you might very well automate tomorrow. Functions decompose into subfunctions.

Environment Box - a source or destination outside the boundary of the domain under study. An environment could be the government, a customer, or other parts of this enterprise.

Flow Line - a flow of material or information between functions or between environments and functions. Flow lines always connect a box at each end. They never split, join or remain unattached.

Flows can connect in four ways to functions: input, control, means, and output.



(Figure 7. Functional Decomposition.)

Input Flows are consumed by the function to create the output.

Output Flows are produced by the function.

Control Flows affect or direct the processing of the function.

Means Flows are the mechanisms which support the function.

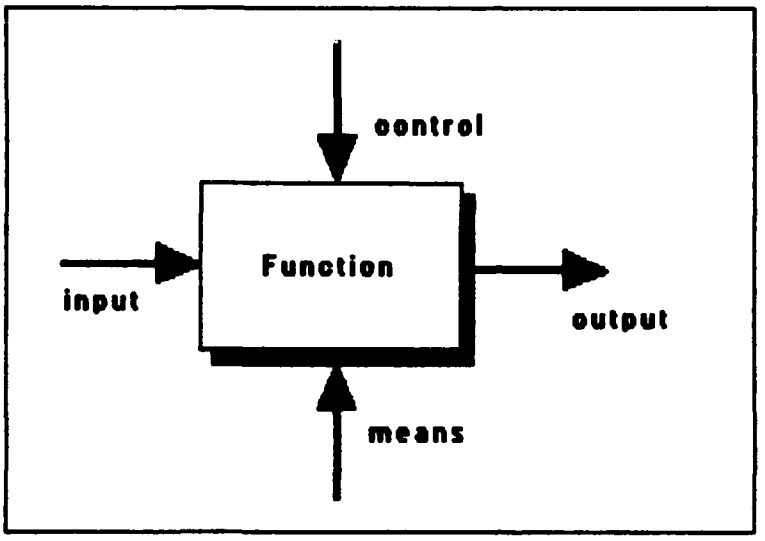
Means can be material (such as a group of persons or a tool) or immaterial (such as a body of knowledge or philosophy). It is important that any means documented in a functional model be truly necessary and not just the part of the current situation.

There are three kinds of flows:

Material Flows

Information Flows

Energy Flows



(Figure 8. Types of Flows.)

3.6.1.3 Functional Flow Abstractions.

The Functional Flow Model contains three abstractions:

1. Abstraction from detail. The hierarchy of composition, decomposition abstracts from the bottom level details.

2. Abstraction from physical mechanisms. The activities modeled are truly functional. They contain no details of manual or automated implementation.

3. Abstraction from time. The flows do not start or stop. They flow constantly. There are no stores for delaying and batching the flows.

3.6.1.4 How to Construct a Functional Flow Model.

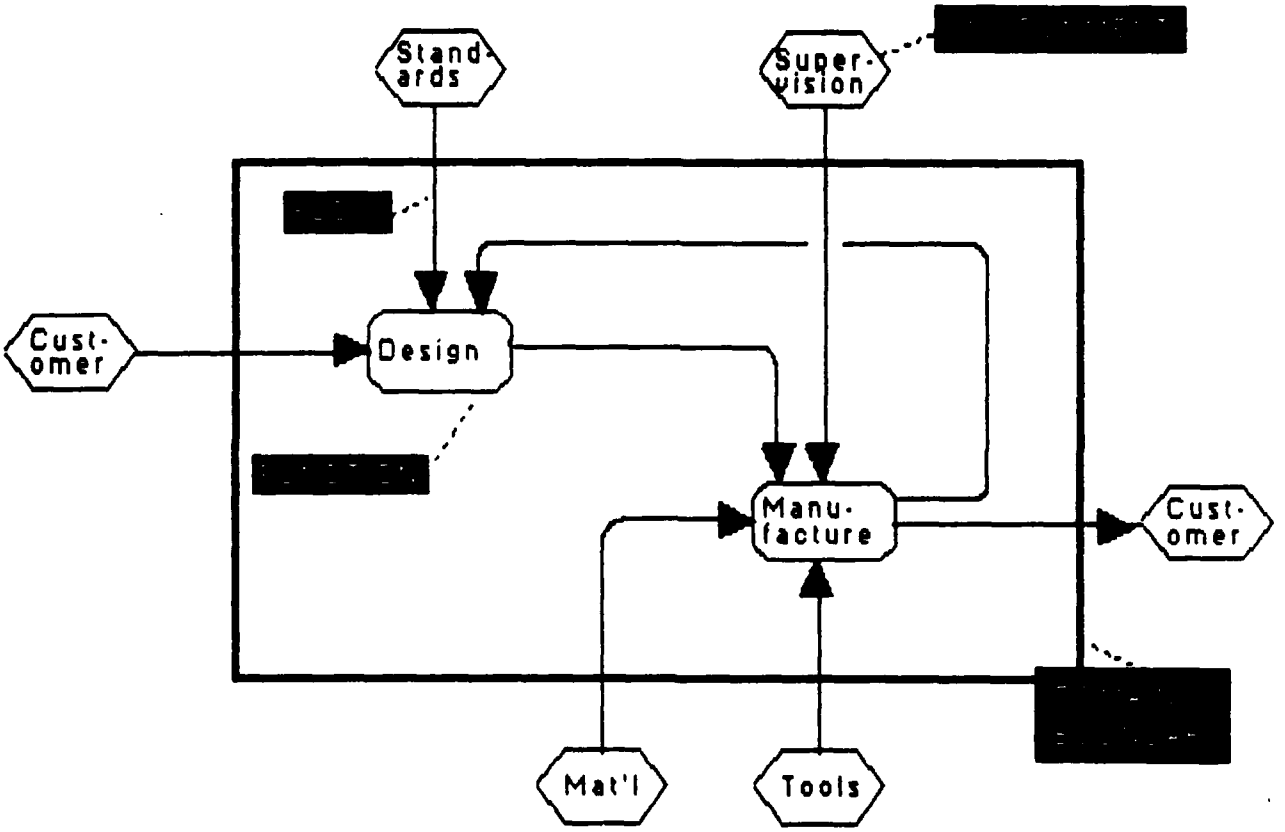
How do you make a functional flow model? We've tried several ways before concluding that the following steps work best:

1. Do a top-down decomposition of the functions.

Why? Why would you do a top-down decomposition of the functions when what you're really interested in is the flow model?

The answer is that functions are the heart of the flow model. You have to get them right first.

Analysts and users often change their minds on the decomposition as they learn more. It's easy to change a decomposition. It's hard to change a flow model.



(Figure 9. Functional Flow Model Concepts.)

Decomposition goes faster than a flow model.

Follow-on efforts often concentrate on only part of the model. Avoiding flow models for these out-of-scope parts saves time.

Analysts must put much mental effort and many hours of diagramming into making flow models. There is a natural reluctance to change the model even if they find a better functional decomposition.

2. Select the level for making the flow model.

It's not necessary or desirable to make a flow model for every level of decomposition. Choose carefully.

3. Do a bottom-up construction of the flows.

Why does flow modeling work best when made bottom-up?

Bottom level functions are small units whose behavior is specific and obvious. If the model builder needs to determine the flows, he can go to the specific person or group responsible for the function. At the high level, the responsibility for functions is more diffuse. It is difficult to find who can help.

Bottom level in-flows and out-flows are more specific. Higher level flow contents are necessarily more vague.

You can compose flows and functions mechanically because the parts identified. You will find it more difficult to decompose because the parts are unknown.

Theoreticians recommend flow models with 5-9 functions per piece. We don't. Our practical experience shows that many analysts need and can absorb diagrams of 20 or more functions. The detail of these diagrams is necessary to their understanding. An artificial limitation only increases ambiguity.

4. Iterate as necessary.

'Nuf' said.

3.6.2 Binary Semantic Models. 8,9

3.6.2.1 The Purpose of a Binary Semantic Model.

A binary semantic model represents all object types, fact types and integrity constraints of the domain. The binary model documents the meaning of the data in engineering, manufacturing and management functions. You have to agree upon meaning for unambiguous exchange and sharing of common concepts and ideas.

The PRECISE * binary semantic model is the same as the ISO binary relationship model.

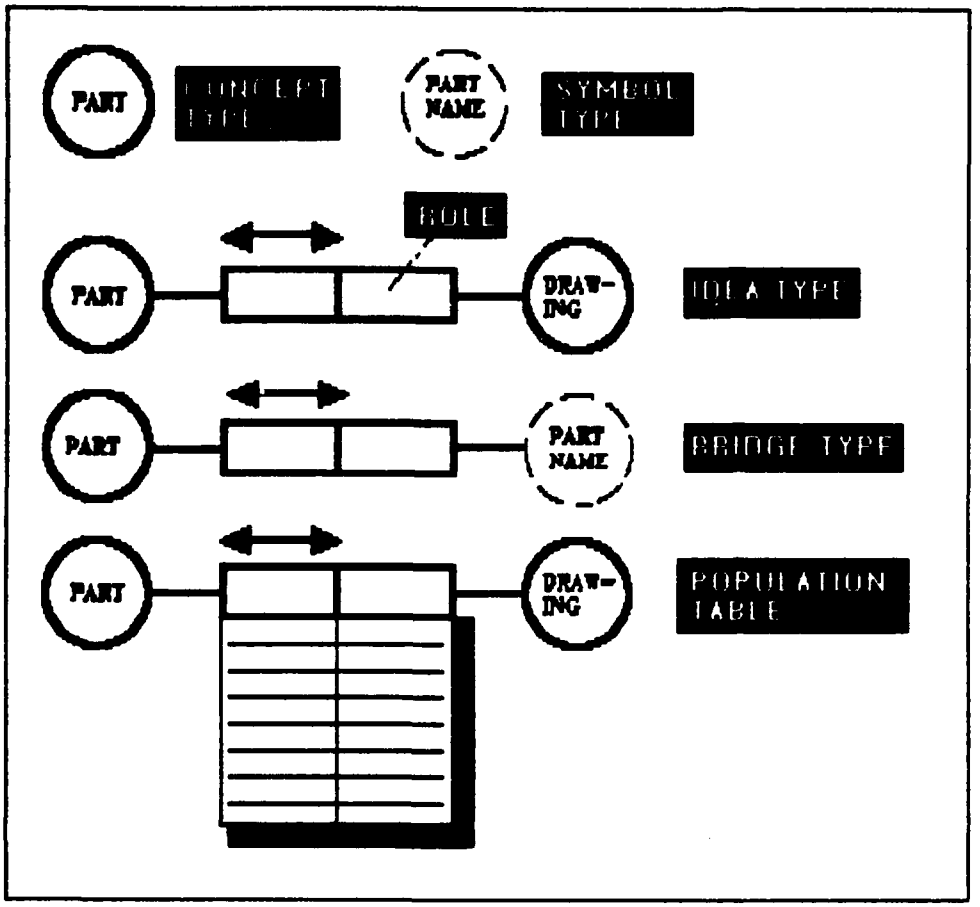
3.6.2.2 Binary Semantic Modeling Concepts.

Object Type - person, place, object, thing, event, etc.

Solid Circle - represents a **Concept Type**: the intensional definition of any "real world" object or event. Example: PART.

Dashed Circle - represents a **Symbol Type**: a set of names, numbers, codes or descriptions which REFER TO the concepts. Example: PART-NUMBER and PART-NAME.

Often there is no one-to-one relationship between Concept Types and Symbol Types. You will find some concepts with more than one symbol used to refer to them (for example PART-NAME and PART-NUMBER). Some Concept Types have no officially recognized Symbol Types (for example, a re-mount corrective operation in the manufacturing process may not have an identification).

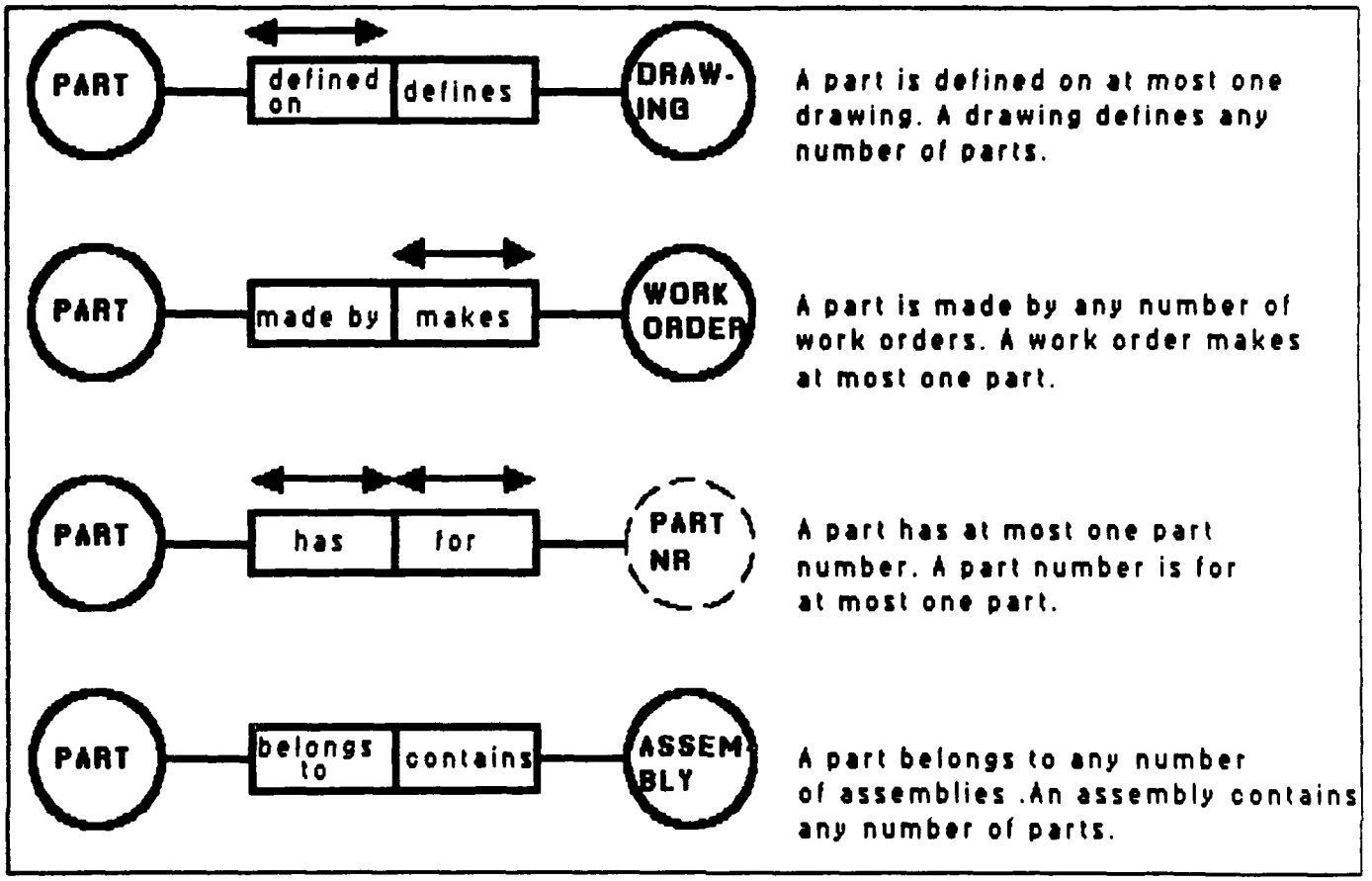


(Figure 10. Binary Semantic Model Concepts.)

Fact Type - a named specific relationship type between two object types.

An **Idea Type** - is an information bearing relationship between two Concept Types. Each concept plays a role in the relationship. The roles go in the boxes next to the corresponding Concept Types. The presence of both roles permits reading the relationship in either direction.

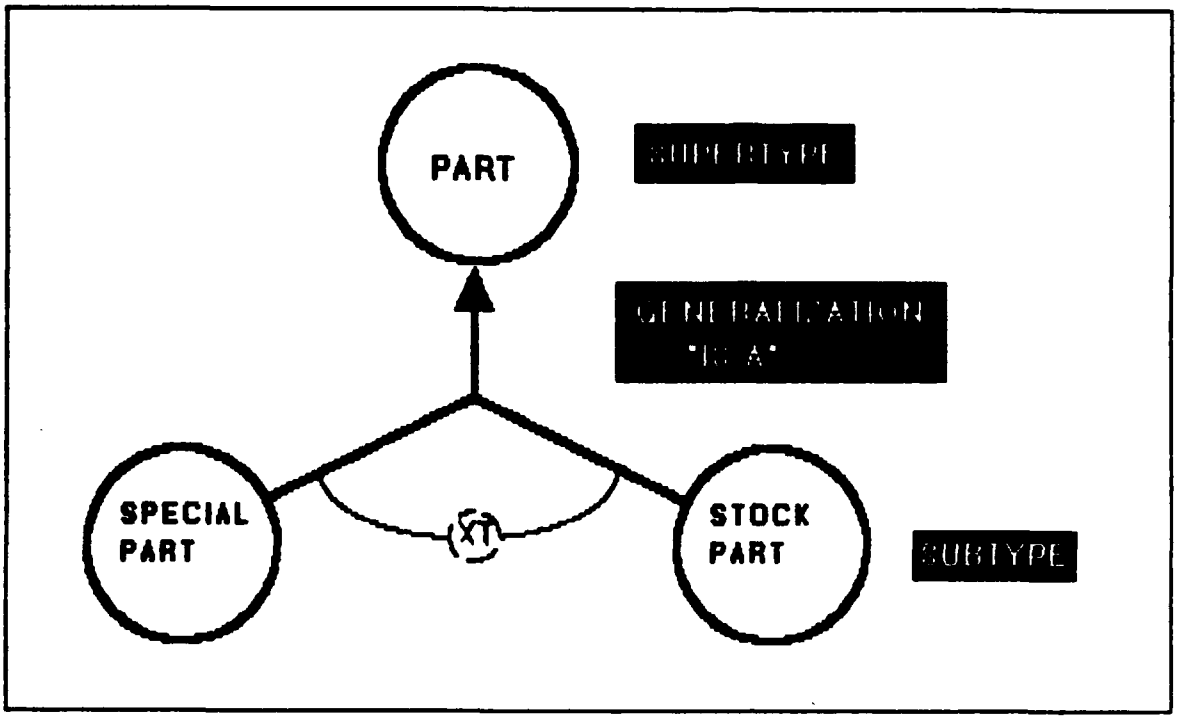
A **Bridge Type** - is a naming relationship connecting a Concept Type with a Symbol Type. We call it a Bridge Type because it bridges between the concepts and the representations for these concepts.



(Figure 11. Simple Uniqueness Constraints.)

Subtype - subtype and supertype relationship. Two solid concept circles connected by a straight line with an arrowhead. The arrow points to the supertype. For example, a QUALITY-TEST is a subtype of a TEST.

Constraints - restriction rules on the information structure states or transitions to preserve integrity. Constraints are formal knowledge rules about natural laws or enterprise policy. We show constraints with dashed lines.

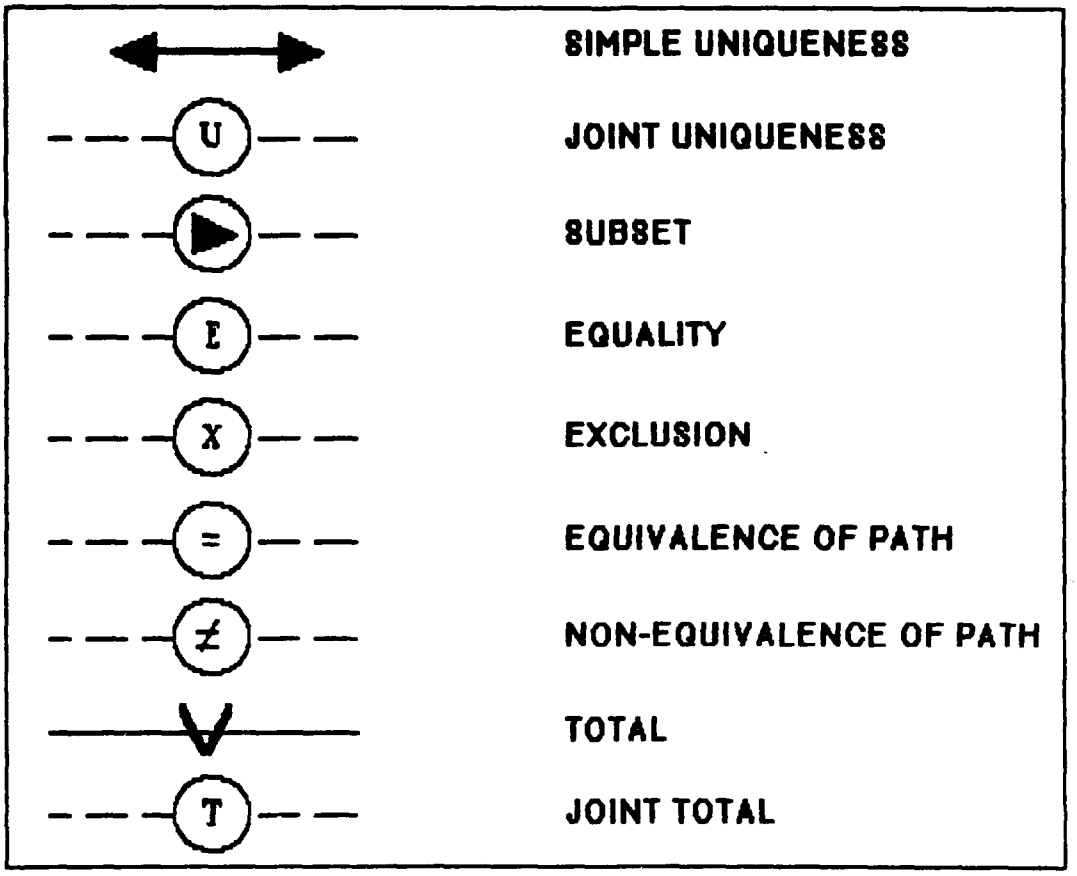


(Figure 12. Subtype Notation.)

For example, there may be an enterprise rule that you must officially release a part before you manufacture it. This is a constraint. Likewise, there is a "law" of nature that a person may have only one age. This too is a constraint.

An intelligent database management system enforces these constraints when programs add to or delete information from the database. It uses these constraints to guarantee the quality, usefulness, and consistency of the information. With constraint enforcement, no user can accidentally or deliberately introduce nonsensical or inconsistent information.

In traditional systems, constraint enforcement has been undocumented, non formal, and haphazard. Lack of constraint checking has greatly increased the cost of traditional development and use.



(Figure 13. Constraint Notation.)

A few commonly occurring constraint types are:

Subtype Exclusion - two or more subtypes are mutually exclusive.

Subtype Total - the union of two or more subtypes equals the total supertype.

Role Uniqueness - a role or combination of roles is unique.

Role Subset - the population of the objects involved in role1 must be a subset of the population of the objects involved in role2. Role subset specifies precedence.

Equivalence of Path - two different paths from object type 1 to object type 2 give the same result.

3.6.2.3 Binary Semantic Model Abstractions.

1. Abstraction from instance to type. (Object Types and Fact Types.)

2. Abstraction from name to concept. (Concept Types versus Symbol Types and more involved references.)

3. Abstraction from particular viewpoints. All viewpoints are equally valid. There is no one entity preferred over another. Normalization does not impose a database technology viewpoint.

4. Abstraction from data oriented structures. (Symbolic, not data reasoning.) Also every valid conceptual access path is modeled, and only valid paths!

5. Abstraction from specific to general. The generalization hierarchy (subtype-supertype) permits the specification of generalization relationship and the inheritance of supertype properties.

3.6.2.4 How to Construct a Binary Semantic Model.

You construct a binary semantic model bottom-up or top-down or middle-out. Usually you build it with some combination of all three approaches.

You don't have to find the elementary pieces of information in any particular way. You can use them whenever you find them. It doesn't matter where or when or how. You have this flexibility because the model can handle elementary concepts. It doesn't force you to take design decisions first.

On the other hand, suppose you wish to build your model more systematically. Large project demand this. Not to worry. There are many methods and they all work. You can, for example, trace and analyze a flow through the functional flow model. Or you can concentrate on the information needed by just one function. Or you can use another method.

You build the binary semantic model graphically, using large sheets of paper. (Interactive graphics tools are now beginning to appear). First draw objects and their subtype connections. Then connect ideas and bridges to the objects. Finally draw constraints connecting objects and roles. As you construct the diagram, you constantly check its correctness and meaning by verbalizing the symbols in English and by using population tables.

Interviewing is the most common technique people use in the initial stages of building a binary semantic model. Users are at ease

explaining in English the information they need, the objects of their environment and their rules of information behavior. Later in the analysis users begin to understand the symbolic language of the binary semantic model diagram and want to help make it. Users can learn the technique in a training course of only a few days.

During the initial stages a professional Information Analyst is very helpful in guiding user discussions, keeping them focused on the subject and explaining the techniques.

4.0 SUMMARY.

4.1 Conclusions.

The increasing use of automation in the manufacturing industries is causing the development of Information Engineering techniques. Techniques such as building models, and especially building the right model, are now a major activity of information engineers who are helping companies in the transition from the industrial to the information age.

Standards bodies have done a good work in developing high level, conceptual models: the Open System Interconnection 7-Layer model, the ANSI/SPARC 3-Schema, and the ISO Conceptual Schema models.

Information Engineering Frameworks and practical guide models for applying Information Engineering models incorporate Reference Models. The PRECISE * Information Engineering framework model illustrates one vendor's mature approach to effective delivery of Information Engineering in the commercial environment.

The two technical models discussed (Functional and Binary Semantic Models) are very formal conceptual modeling techniques. Both find a place in the PRECISE * IE framework. The Binary Model also is in the ISO Reference Model.

Project experience shows the use of Information Engineering techniques is improving the productivity and quality of information systems development.

4.2 Future Directions.

Predicting the future is always a little risky - as they say, it's much safer to predict the past... However, we will attempt to make some predictions based on many years' work in this field.

First, there will continue to be increasing acceptance of Information Engineering work. Fewer people will question the

necessity and expense of building models BEFORE implementation and value of capturing high level specification knowledge.

Second, we will develop tools to capture and analyze models, to present them lexically and graphically, and to generate designs and implementations directly from the models. We'll make these tools graphically oriented for maximum communication value. Also, we'll design these tools with the very methods they support.

Third, Information Engineering will draw on ideas and tools from Artificial Intelligence. IE practitioners will come to recognize that rule based knowledge is as important as functional and information knowledge and will develop or adapt methods to model rules. They will begin to use AI tools such as object oriented programming and object oriented databases and AI Workstations. The IE tool builders will use AI techniques in their tools.

In turn, the AI researchers will realize the need for a Knowledge Engineering Methodology to build large scale expert systems. The will eventually see that Information Engineering has the technical models for domain analysis and the framework concepts for handling large projects. They will adopt many information engineering models as part of the techniques of knowledge acquisition.

We are just at the start of a revolution in the manufacturing industry and in society. It promises to be an interesting experience.

5.0 ACKNOWLEDGEMENTS.

I have benefited greatly from design meetings and joint work with the members of Control Data's Information Engineering Technology Center: Joan Milloy, Denny Mikkelson, Jon Stonecash, Jim Anderson, Roger Thyr and Necito Delacruz; and from in depth discussions with Dr. Sjir Nijssen, Dr. Robert Meersman, Frans van Assche and Baba Piprani; and from many, many customers and students.

ENDNOTES

1. "Information Processing Systems,
Open Systems Interconnection Basic Reference Model,"
American National Standards Institute,
1430 Broadway,
New York, NY 10018

2. "Critical Issues in Integrating Factory Automation Systems,"
Nicholas G. Odrey and Roger Nagel,
CIM REVIEW, volume 2, number 2, Winter 1986

3. "A LAN Primer,"
Dick Lefkon,
Byte Magazine, pp. 147-154,
July 1987

4. "Interim Report,"
The ANSI/SPARC Study Group on
Data Base Management Systems,
1975

5. The ANSI/X3/SPARC DBMS Framework Report
of the Study Group on Database Management Systems,
AFIPS, 1977

6. "Concepts and Terminology for the Conceptual Schema and
Information Base,"
ISO TC97/SC5/WG3
Van Griethuysen, (ed.)
1982
(ANSI publication number N197)

7. "Assessment Guidelines for Conceptual Schema Language
Proposals,"
ISO TC97/SC21/WG5-3
Van Griethuysen and King, (editors.)
1985

8. "Natural Language Analysis, Information Modeling and Database
Engineering"
Paul Thompson
Eighteenth Annual Hawaii Systems Conference
January 1985

9. Informatie Analyse volgens NIAM in Theorie en Praktijk,
J.J. van Wintraecken,
Control Data and Academic Press,
(English edition 2th Q 1988)

Appendix

10. PRECISE * Information Engineering Overview,
Publication Nr. 76070010,
Control Data Corporation,
P.O. Box 0, HQB01D,
Minneapolis, MN 55440
1-800-828-8001 ext 2104
1986

11. PRECISE * Information Engineering Textbook,
Control Data Corporation,
Information Engineering Technology Center (IETC),
January, 1987

Appendix

ENTITY LINK KEY ATTRIBUTE SEMANTIC INFORMATION MODELING: THE ELKA MODEL

Timothy L. Ramey
Robert R. Brown

Contributed by Robert R. Brown

ENTITY LINK KEY ATTRIBUTE SEMANTIC INFORMATION MODELING

The ELKA Method

Timothy L. Ramey
Robert R. Brown

1 Background

The Entity Link Key Attribute (ELKA) information modeling technique is a means of capturing an understanding of the information with which a system deals. This technique consists of a theory of application, called the method, and a practice of application, called the discipline. The method defines a means of describing a system in terms of the information that is observed to be utilized by that system. It is a theory for capturing a representation of the "real world" having sufficient semantic content to be meaningful during system operation. The method is, however, only a way of looking at the world; it contains no language for expressing what is observed. The discipline introduces both a language in which these observations can be communicated and a process by which organizations of user personnel can build viable ELKA information models.

This paper presents an overview of the ELKA method, which was initially developed in 1978–1979 and saw gradual enhancement through 1982. While a low level of activity concerning its development continues today, this paper concentrates on the state of ELKA as it was developed in that period. The discussion has been couched in relatively straightforward terms to make it more accessible. However, it should be read with caution — most of the concepts discussed have fundamental and far-reaching impact on the ELKA technique. An understanding of the ELKA method is not required of users of the ELKA discipline and may even be viewed as detrimental for some such users. Some of the continuing issues with the ELKA technique are discussed in the Continuing Issues section. Relevant references to ELKA, its counterpart IDEF$_1$[1] and other information and data modeling approaches are listed in the Bibliography section at the end of this paper.

[1] IDEF$_1$ is a technique based on ELKA developed for the US Air Force in this same period to help aerospace manufacturers analyze their factory operations. It is not related to a commercial database modeling tool being marketed under the name IDEF1X.

2 Basis

The problem of information modeling can be broken down into several issues that can be addressed separately. One of the first issues to be raised is that of "modeling." What do we intend by producing a model? In the ELKA modeling method, a model is seen as a representation of the real world as measured by an observer. This belies the particular ontology[2] that is accepted in the ELKA method. The model is observer-based. That is, while there may (or may not) be an underlying reality, the ELKA method addresses *only* the observed reality.

Several problems are raised in observer-based systems of ontology that must be resolved. These issues are no different than those traditionally faced in the philosophy of science, and it seems quite reasonable for the ELKA method to adopt the most common solution to them. Therefore, in the ELKA method the dictum of "consensual validation" is applied. Taken literally, this means that the consensus of observations is used to define the model that represents those observations. This resolves the problem of two observers reporting different realities by accepting a consensus view of reality (essentially, calling in a "neutral third party" to resolve the dispute). The problem posed by one observer reporting a single observation while another reports two is more subtle but is resolved as well.[3] In this case the observer may be considered as an instrument that measures the real world. The precision and sensitivity of one such instrument should not be expected to match that of any other instrument. Therefore, the disparity of observations is resolved in terms of the precision and sensitivity of instruments. By consensual determination, the most appropriate instrumentation (and report) is selected.

Another ontological issue that presents itself is that of how the observed world is to be divided up. Such subdivision is fundamental to modeling and analysis because it allows analysts to group and compare similar component parts of various observations. This ultimately leads to the statement of principles that involve *types* of such component parts. In fact, the grouping together of component parts in such a way that statements can be made about component parts being instances of one type as opposed to another type is itself an important achievement. In the ELKA method, the first statement of this kind is about how ELKA models fit within the four-space continuum. ELKA takes a deliberately static view

---

[2] Henry Margenau. *The Nature of Physical Reality: A Philosophy of Modern Physics*. New York: McGraw-Hill, 1950.

[3] In fact, it should be apparent that these two cases are synonymous, depending only on the point of view of the observer of observers.

of the world. This is not to say that there is no notion of time in the modeling method but rather that a particular model is viewed as representing a viable slice through four-space in which stability of the model across the time dimension of the slice is quite high. There is no provision in the method for model dynamics. Hence, while it may be that different models are appropriate at different locations in four-space, each individual model is viewed as independently viable and stable.

In the ELKA method a mapping is assumed between the information that is observed and the articles in the real world that are the subject of observation. It is not a part of the theory of ELKA to resolve this mapping, only to resolve the various observations into a semantically viable representation of the overall observed world. This, of course, introduces a new issue for resolution: the universe of discourse must be determined. Since each instrument through which the world is measured differs slightly in precision and sensitivity, it may be expected that the extent of any universe of discourse will be directly related to the number of instruments (that is, observers) employed. This issue is actually more complex, however, since many different mappings of reality will occur for each observation and since it is likely that many (if not most) of these mappings will fall outside the intent of any particular modeling effort. The notions of *purpose* and *viewpoint* in modeling are introduced to limit the scope of any particular modeling effort. The *purpose* for modeling reflects the objectives of the effort, essentially acting as a filter on observations. The *viewpoint* of the model (or modeler) delimits the initial point of reference in modeling, essentially limiting the focus of each instrument. Finally, for each model, the *context* of the model is established, identifying that portion of four-space in which the model will have viability. The ELKA method defines the requirement for these statements but, as with other aspects of the modeling technique, prescribes no mechanisms for their realization.

The ELKA method actually deals with three aspects of information. The *existence* (observation) of information is dealt with in the method by the naming and defining of observed properties within the problem domain. This task translates into the major concentration of effort within the ELKA discipline. With the existence of information elements established, the method deals with the *structure* of information by establishing distinct groupings and inter-grouping associations. These structural characteristics of a model represent both the abstraction of information elements into general categories and the establishment of certain existential rules that apply to the information represented. Finally, the method deals with the *identity* of information by the establishment of mechanisms within a model that provide

for the unique identification of individual members of structural groupings. While the first two aspects (existence and structure) are sufficient for an abstract understanding of the information modeled, this latter aspect (identity) is crucial to the communication of information. Without the ability to uniquely indicate a particular element of information within the total context of an observed universe of discourse, it is not possible to communicate about the observed information in any but abstract terms.

The ELKA method also divides the observed world into units of observed information called *attributes*. These are the individual measurements provided by the instrument/observers as they record the real world. An attribute is represented as a coupled pair consisting of a **name** and a **value**. Hence, the following pairs might be observed:

weight	165 lbs
color	grey
first name	Frank
time	2137
unit price	$7132.00
authorization	03426
issue date	810307

Further, certain separately observed units are identical in their underlying nature. For example, when we observe

```
manager "John Williams"
```

we infer from this the information

```
employee-name "John Williams"
```

For this reason, the ELKA method establishes two naming conventions for information elements:

a. attribute name — agreed to be the most fundamental naming convention for the information

b. attribute tag — a naming convention for the *use* of the information in some specific context.

This corresponds to the general notion of denotation and connotation, in that the attribute name is viewed as representing the denotative naming application of the information, while the various attribute tags that may correspond to it are viewed as representing connotative applications of that information.

These units of observed information are clustered together into abstractions called *entities*. In fact, while it is the units (attributes) that are observed, it is these abstractions (entities) that are most frequently referred to in everyday life. For example, the sentence "My car gets 23 miles per gallon ..." associates an observed measurement (23 miles per gallon) with a particular cluster of information (my car). This cluster is used in referring to an entire cluster of information as if it were a single unit. Another kind of clustering occurs when several such entities are associated with each other. In the ELKA method, one of the ways this can happen is through what are called *links*. The previous example contains such a link: "my car" associates an individual person (me) with another observed object (car). Such links in ELKA are always binary — they associate one cluster of information with another cluster of information. However, a given entity may take part in any number of such binary associations.

For ease of reference and discourse, these elements (attributes, entities and links) are grouped together into *types*. A type is essentially a rule structure that defines the attributes, entities or links that can be instances of it. Since a model is only considered to have viability within some particular segment of four-space (that is, it is not considered to be "universally viable"), this analysis is extended slightly to *classes* that are made up of the existing instances of particular types.[4] Hence, while a given type may be viewed as defining an infinity of instances, an associated class may be viewed as having a finite membership. So it is that in ELKA, information models are composed of classes of attributes, classes of entities and classes of links. A particular attribute class, for example, has as members the denotations of measured information of a given type (*eg.* all first names that are known [currently exist] within a system). Finally, in order to distinguish one member of an entity class from the other members of that entity class, the ELKA method calls for the concept of a key — which is a set of attributes used to create such a distinction — and the associated concept of a key class.

3 Formalism

The following is a description of the ELKA method in terms of its own descriptors.[5]

[4] The notion of "class" in this sense may be considered to be similar to the notion of "set" in mathematics. Types are referred to as having instances, while classes are referred to as having members. By definition, a member of a class is also an instance of the type that defines the class.

[5] This is a "formalism" only in the "informal" sense. A true (that is rigorous and well-founded — not to mention, remarkably more elegant) formalism is currently being developed by Dr. Chris Menzel at Texas A&M University as part of a general study of the semantics of commonly used modeling

The symbology used is arbitrary and should be interpreted in the following way:

$$EC[ac_1, ac_2, \ldots, ac_n]$$

represents an entity class (EC) whose member entities each represent information that is indicated as $ac_1, \ldots, ac_n$.

$$EC.ac$$

represents the information indicated by ac in the members of the entity class EC. For example

$$PERSON[name, phone\ number, address]$$

indicates that there is an entity class called $PERSON$ that has member entities that each represent information called *name*, *phone number* and *address*. The *name* information in all $PERSON$ entities would be represented as

$$PERSON.name$$

Link classes are represented by noting the dependence of information appearing in members of one entity class on information appearing in members of another entity class. The notation

$$CAR[owner] \subseteq PERSON[name]$$

indicates that the information called $CAR.owner$ in each member of the entity class CAR must correspond to some information called $PERSON.name$ in a member of the entity class $PERSON$. Stated more formally, it says that the information labeled $CAR.owner$ defines a set that must be a subset of the set defined by the information labeled $PERSON.name$. This also extends to groupings of such information, such that the sets may be considered as composed of members formed by the concatenation of the named items of information whose classes are listed in the groups. Hence,

$$CHECKOUT[book, copy]$$
$$\subseteq VOLUME[book_id, copy_number]$$

indicates that the information on a book checkout called *book* and *copy* must also appear as information in the entity class $VOLUME$ correspondingly called *book_id* and *copy_number*.

techniques. Readers who understand the difference should wait patiently for Dr. Menzel's results — in the meantime, however, we offer this "informal" demonstration in the spirit of a "heuristic" mathematical discussion.

These examples would indicate a link class between *PERSON* and *CAR* in the first case and between *VOLUME* and *CHECKOUT* in the second case.

Finally, a notation will be used to indicate what will here be termed semantic inference.[6] The various rules of the ELKA method create a situation in which it is possible to infer from the existence of a member of one entity class the existence of a member of another (or sometimes the same) entity class. Due to the network structure of a model created through this method, it is possible to follow such "inference paths" from one entity class to another through several alternative intermediary inference paths. When this occurs, semantic ambiguity is possible in the model.[7] The notation

$$(CHECKOUT - VOLUME - BOOK)$$

for example, will be used to indicate an inference path from the entity class *CHECKOUT* to the entity class *VOLUME* to the entity class *BOOK*. The notation

$$(PO - VENDOR - EMPLOYEE)$$
$$\equiv (PO - EMPLOYEE)$$

will be used to indicate that the result of the inference path from *PO* (purchase order) to *EMPLOYEE* via *VENDOR* (that is, a member of the entity class *EMPLOYEE*) must be the same as the result of the inference path from *PO* directly to *EMPLOYEE* (also a member of the entity class *EMPLOYEE*).

An entity is something described within the problem domain of the model. A collection of entities that would be described in a similar manner is called an entity class (*EC*). That is, entities that represent the same types of measurement are grouped together into an entity class. Entity classes are given entity class names (*ecn*) by which any particular entity class within a model may be identified. Hence, the method defines

$$EC[ecn] \tag{1}$$

where

$$\forall EC[ecn]_i : \\ (\neg\exists EC[ecn]_j \mid (EC[ecn]_i = EC[ecn]_j) \wedge (i \neq j)) \tag{2}$$

---

[6] In some *data* modeling techniques (particularly those based on, or closely related to, the CODASYL model) these semantic inferences correspond quite well to the notion of navigation through a data model of a database.

[7] The section **Continuing Issues** discusses the problem of semantic ambiguity due to inference paths. An *ad hoc* solution is used in this report that appears to have some merit.

That is, all *ecn* must be unique within an particular model.

An attribute represents a property recorded about an entity. An attribute class (*AC*) is a collection of similar properties. An attribute class is said to be owned by an entity class in the sense that denotation of a particular type of information is determined by its defining environment. Each attribute class is given an attribute class name (*acn*). The method defines

$$AC[owner, acn] \tag{3}$$

where

$$\forall AC[owner, acn]_i : \\ (\neg\exists AC[owner, acn]_j \mid (AC[owner, acn]_i = AC[owner, acn]_j) \wedge (i \neq j)) \tag{4}$$

and

$$AC[owner] \subseteq EC[ecn] \tag{5}$$

That is, *AC.owner* is a connotative use of *EC.ecn* and is said to have been inherited from *EC* by *AC* due to a link class existing between them. Hence, for any member of *AC*, there must be some member of *EC* such that *AC.owner* ≡ *EC.ecn*.

It will be noted that the attribute class group in AC must be a subset of the same group under EC that was listed as being unique. That is, *EC*[*ecn*] from rule (5) is the same as *EC*[*ecn*] from rule (2). The reason for this will be seen in rules (24) through (31), which deal with the dependency of attribute uses associated with one entity class on attribute uses associated with another entity class. The implication is that, while an entity class (*EC*) may be related to many attribute classes (*AC*), each attribute class must be related to *exactly one* entity class.

Not illustrated in this discussion is the means by which the basic model is extended to include, or cover, the members of each of the classes shown. This extension would illustrate two additional rules that will be stated informally. The first is that a given attribute class is only allowed to represent a single attribute at a time within a given entity. This is often called the "no repeating value" rule in that it forces the model to represent repeating groups of values as separate entities (in entity classes often created in response to discovery of such cases). The entity class *AC* is actually an example of this, in that each entity class (*EC*) might have an attribute class called *ATTRCLASS* containing images of all of the properties it represents. However, each member of *ATTRCLASS* (appearing in one member of *EC*) would have to have many members (attributes) at a time to accommodate all of the properties

defined by that entity (the member of EC). As a result, a separate entity class (called AC) is created to represent all the attribute classes that might occur in a model.

The second rule not formally stated here is that the model only represents assertions that are viable under the same circumstances under which the model is itself viewed as being viable. This means that each attribute class that is shown in a model must represent attributes that will always be present in their associated entity class members. For example, this rule requires that there must always be an attribute associated with the attribute class *ecn* for every member of the entity class EC. This rule is often called the "no null value" rule in that the model may not represent situations where the value associated with an attribute from an attribute class would, under some circumstances, be "null" or undefined. The creation of entity classes that represent the "sometimes present" aspect of these null-value situations is dictated by this rule.

An attribute use class (AUC) is the referenceable, or "visible," use of an attribute class to record measurements in the members of an entity class.[6] Each attribute use class is given a usage name (*tag*), which is the connotative naming convention for that information in that environment (*ie*, in the particular juxtaposition provided by the entity class in which it is displayed). The method defines

$$AUC[user.tag, owner.acn] \tag{6}$$

$$\forall AUC[user.tag]_i : \\ (\neg\exists AUC[user.tag]_j \mid (AUC[user.tag]_i = AUC[user.tag]_j) \wedge (i \neq j)) \tag{7}$$

$$AUC[user] \subseteq EC[ecn] \tag{8}$$

$$AUC[owner.acn] \subseteq AC[owner.acn] \tag{9}$$

A key is the means by which one member of an entity class is distinguishable from the other members of that entity class. Such a key is actually a set of attribute uses. A key class (KC) is a collection of like keys for a particular entity class. This defines, in a sense, a double-edged sword in that a collection of attributes that may be used to distinguish members of an entity class must, of necessity, be unique (when taken together) among all of the members of that entity class. Hence, the key class represents both the more commonly thought of access

[6] The apparent awkwardness of the notation "attribute use" is unfortunate, but it is accurate. This notation points out the fact that it is the connotations that are being discussed. While the attributes (denotations) may be thought of as being shared, it is the attribute uses (connotations) that are dependent upon each other.

path, as well as a requirement for uniqueness. Key classes are given arbitrary identifiers (*kcid*) to provide for their "distinguishability." Hence,

$$KC[owner, kcid] \tag{10}$$

$$\forall KC[owner, kcid]_i : \\ (\neg\exists KC[owner, kcid]_j \mid (KC[owner, kcid]_i = KC[owner, kcid]_j) \wedge (i \neq j)) \tag{11}$$

$$KC[owner] \subseteq EC[ecn] \tag{12}$$

As stated, keys are collections of attribute uses and so the method defines

$$AUCKC[kcowner, kcid, aucuser, tag] \tag{13}$$

$$\forall AUCKC[kcowner, kcid, aucuser, tag]_i : \\ (\neg\exists(AUCKC[kcowner, kcid, aucuser, tag]_j \mid \\ (AUCKC[kcowner, kcid, aucuser, tag]_i = AUCKC[kcownerkcid, aucuser, tag]_j) \wedge (i \neq j)) \tag{14}$$

$$AUCKC[kcowner, kcid] \subseteq KC[owner, kcid] \tag{15}$$

$$AUCKC[acuser, tag] \subseteq AUC[user, tag] \tag{16}$$

$$(AUCKC - AUC - EC).ecn \equiv (AUCKC - KC - EC).ecn \tag{17}$$

Rule (17) states that the entity class that displays an attribute use class as a part of a key class must be the same as the entity class that owns the key class in which this attribute use class appears.

A link is a binary association between two entities (or in certain obscure cases, between an entity and itself). A link class (*LC*) is a collection of similar links. Link classes are said to be binary and are said to have an independent and a dependent 'end.' The modeling method requires that all link classes be resolved into a form that allows for this independent / dependent identification.[9] The entity class on the dependent end of a link class is dependent on the entity class on the independent end of that link class to the extent that there must be a member at the independent end for each member at the dependent end. Each link class has an identifier (*lcid*) that is unique when combined with the names of the dependent and independent entity classes it associates.

[9] The ELKA discipline provides for this resolution to occur gradually as modeling participants become more expert in their understanding of the domain being modeled, the discipline for modeling and the modeling method.

The method defines

$$LC[dep, indep, lcid, kcowner, kcid, card] \tag{18}$$

$$\forall LC[dep, indep, lcid]_i : \\ (\neg\exists LC[dep, indep, lcid]_j \mid (LC[dep, indep, lcid]_i = LC[dep, indep, lcid]_j) \wedge (i \neq j)) \tag{19}$$

$$LC[dep] \subseteq EC[ecn] \tag{20}$$

$$LC[indep] \subseteq EC[ecn] \tag{21}$$

$$LC[kcowner, kcid] \subseteq KC[owner, kcid] \tag{22}$$

$$(LC - KC - EC).ecn \equiv (LC[indep] - EC).ecn \tag{23}$$

Link classes represent the existential constraints that tie entities in one class with entities in another class.[10] Three forms of link class are supported, identified in (18) as *card* (loosely, the cardinality of the indicated association):

a. *one-to-one*, in which each member of the dependent entity class is existentially dependent on one member of the independent entity class and each member of that independent entity class may have either zero or one associated member of the dependent entity class;

b. *one-to-many*, in which each member of the dependent entity class is existentially dependent on one member of the independent entity class and each member of that independent entity class may have zero, one or more associate members of the dependent entity class;

c. *one-to-many (closed)*, in which each member of the dependent entity class is existentially dependent on one member of the independent entity class and each member of that independent entity class must have one, and may have more, associated members of the dependent entity class.[11]

The semantics of a link class are said to be clarified, or (in a sense) instantiated, through the association of a key class of the independent entity class with the link class. Recall that an entity class may have many key classes ((10)-(12)). The semantics of the dependence

[10] As shown in (18)-(23), an entity class may be dependent on itself through one or more link classes. This presents a problem that is discussed under the heading of **Loops and Cycles** in the section **Continuing Issues**.

[11] The "one-to-many" forms are also sometimes referred to as "weak one-to-many" and "strong one-to-many."

between two entity classes that is illustrated by the assertion of a link class between them are expressed in terms of the sharing of information that occurs between the two entity classes. This sharing is realized in ELKA through the appearance of the same attribute use classes in both entity classes (see the following discussion and rules (24)-(31)). In order to assert the semantics we have in mind for any particular link class, then, we have to associate one of the key classes in the independent entity class with that link class. A key class is used because it is precisely this information that must be "shared" by the two entity classes to realize these semantics. Hence we show, in rule (22), that each link class records the key class that is associated with it. And in rule (23) we show that the entity class containing the associated key class must be the same entity class that is independent in the link class.

Each link class involves the "migration" of the attribute use classes appearing in one of the key classes of the entity class at its independent end into the entity class at its dependent end. To resolve the migration/inheritance of members of AUC that appear in a member of KC associated with a member of LC, the method defines

$$AUCLC[to_user, to_tag, from_user, from_tag, kcowner, kcid, dep, indep, lcid] \tag{24}$$

$$\forall AUCLC[to_user, to_tag]_i : (\neg\exists AUCLC[to_user, to_tag]_j \mid (AUCLC[to_user, to_tag]_i = AUCLC[to_user, to_tag]_j) \wedge (i \neq j)) \tag{25}$$

$$AUCLC[to_user, to_tag] \subseteq AUC[user, tag] \tag{26}$$

$$AUCLC[kcowner, kcid, from_user, from_tag] \subseteq AUCKC[kcowner, kcid, aucuser, tag] \tag{27}$$

$$AUCLC[dep, indep, lcid] \subseteq LC[dep, indep, lcid] \tag{28}$$

$$(AUCLC - AUC - EC).ecn \equiv (AUCLC - LC[dep] - EC).ecn \tag{29}$$

$$(AUCLC - AUCKC - EC).ecn \equiv (AUCLC - LC[indep] - EC).ecn \tag{30}$$

$$(AUCLC - AUC) \neq (AUCLC - AUCKC - AUC) \tag{31}$$

Rule 29 states that a "migrated" AUC must appear as one of the displayed attribute use classes in the entity class that is dependent in the associated link class (it must migrate to the target). Rule 30 states the inverse, that a "migrated" AUC must (originally) appear as one of the attribute use classes in the entity class that is independent in the associated link class. Rule 31 states that an attribute use class cannot migrate to itself.

4 Continuing Issues

In this section a brief overview of some of the issues that continue to be unresolved in the ELKA technique is presented. The list of issues presented here is certainly not complete, but it does illustrate some of the major concerns that remain. While achievable solutions (though largely *ad hoc*) are available for some of these issues, many others remain out of our grasp at this point in time.

Partially Determined Links

A common circumstance in the development of ELKA models is for one or more link classes to be semantically equivalent. In other words, it often occurs that the dependency illustrated by a link class is only partially represented in the model. These situations are called *partially determined links*, since the model is only partially able to resolve the underlying information structure. An approach to dealing with these circumstances that seems attractive focuses on the semantic equivalence of the link classes involved. Four types of such equivalences are observed:

a. inclusively independent equivalence — an entity in an entity class may be independent of entities in *all* of several entity classes (which are *all* dependent on it) through links in link classes that are semantically equivalent;

b. exclusively independent equivalence — an entity in an entity class may be independent of entities in *only one* of several entity classes (which are each dependent on it) through links in link classes that are semantically equivalent;

c. inclusively dependent equivalence — each entity in an entity class is dependent on one entity in *each* of several entity classes through links in link classes that are semantically equivalent;

d. exclusively dependent equivalence — each entity in an entity class is dependent on an entity in *only one* of several entity classes through links in link classes that are semantically equivalent.

To illustrate, consider the following situation. A publicly held corporation (call it *BIGBUCKS*) issues stock for sale. The issues of stock are subsequently purchased by either a corporation or an individual in each case. BIGBUCKS wishes to know, for tax purposes, each of the owners of its stock. We might begin by asserting that *STOCK* (the entity class

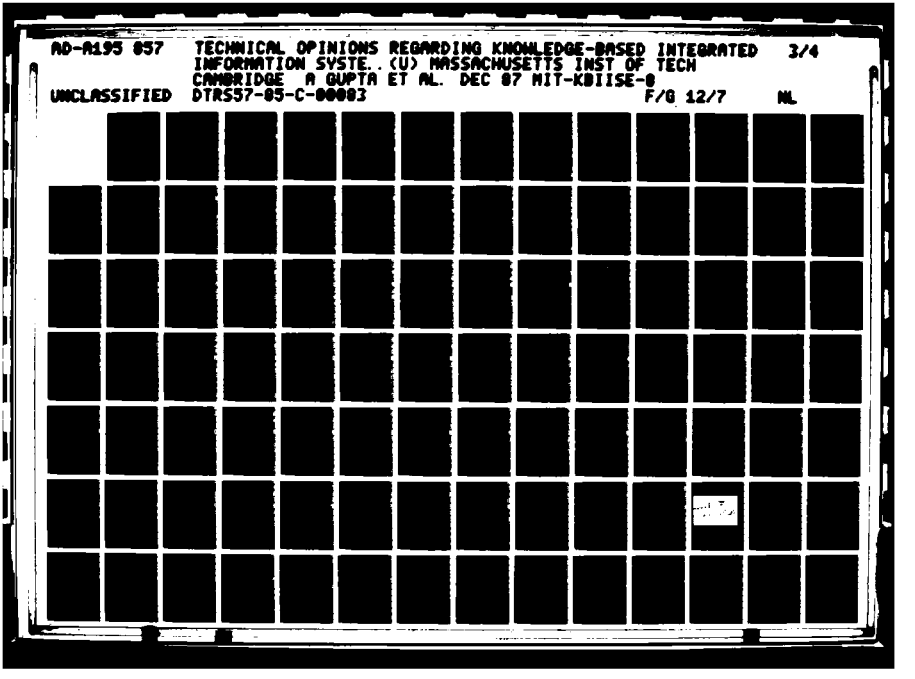
representing all the issues of stock purchased) is dependent in the link class *owns_stock* on *STOCK_HOLDER* (the entity class representing all the owners of issues of stock). However, the membership of *STOCK_HOLDER* could not be homogeneous; that is, some members of *STOCK_HOLDER* would be corporations (with corporate identifiers) and some members would be individuals (with social security numbers). Since the definitions for these two "sub"memberships are not semantically equivalent, we have a problem. One way to deal with this problem might be to give each member of *STOCK_HOLDER* its own identifier (from a homogeneous domain) and then make *STOCK_HOLDER* independent in one-to-one link classes identifying the stock holder as *CORPORATION* or *PERSON*. Here, there would be two difficulties: (1) we have no way of ensuring that all stock holders have associated with them members of *CORPORATION* or *PERSON* and (2) we have no way of ensuring that a stock holder is *either* a corporation *or* a person *but not both*. The solution to this dilemma is to recognize that the ownership of stock by a corporation is semantically equivalent to the ownership of stock by an individual — both of which we would represent as link classes (without intervening entity classes). Hence, we can assert that there is an exclusively dependent link class equivalence between a link class "*owns_stock*" making *STOCK* dependent on *CORPORATION* and a link class "*owns_stock*" making *STOCK* dependent on *PERSON*.

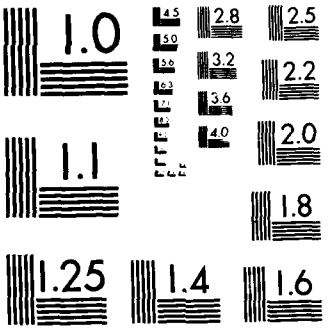
A side effect of this approach is that the key classes that are associated with "dependent" link class equivalences must themselves be equivalent. In the example, if the key class for *CORPORATION* associated with the link class *owns_stock* consisted of the one AUC called *FSCM* (Federal Stock Code for Manufacturers) and the key class for *PERSON* associated with the other link class *owns_stock* consisted of the one AUC called *SSN* (Social Security Number), the assertion that the two link classes are equivalent in the context of *owns_stock* makes it necessary that the two key classes (*FSCM* and *SSN*) behave equivalently — that is that they must be drawn from the same (perhaps composed) domain.

Equivalences have names by which they can be identified (*equivid*). In the previous example, we might use the name *stock_holder_of* as indicative of the shared semantics of the link classes involved. For link class equivalences, we may define the following:

$$EQUIV[equivid, eqtype] \tag{32}$$

$$\forall EQUIV[equivid]_i : \\ (\neg\exists EQUIV[equivid]_j \mid (EQUIV[equivid]_i = EQUIV[equivid]_j) \wedge (i \neq j)) \tag{33}$$


AD-A195 857 TECHNICAL OPINIONS REGARDING KNOWLEDGE-BASED INTEGRATED 3/4
INFORMATION SYSTE..(U) MASSACHUSETTS INST OF TECH
CAMBRIDGE A GUPTA ET AL. DEC 87 MIT-KBIISE-8
UNCLASSIFIED DTRS57-85-C-00083 F/G 12/7 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

$$LCEQ[equivid, dep, indep, lcid] \tag{34}$$

$$\forall LCEQ[dep, indep, lcid]_i : \\ (\neg\exists LCEQ[dep, indep, lcid]_j \mid (LCEQ[dep, indep, lcid]_i = LCEQ[dep, indep, lcid]_j) \wedge (i \neq j)) \tag{35}$$

$$LCEQ[equivid] \subseteq EQUIV[equivid] \tag{36}$$

$$LCEQ[dep, indep, lcid] \subseteq LC[dep, indep, lcid] \tag{37}$$

The effect of *LCEQ* is to assert that each link class may be involved in at most one equivalence. Rule (35) has the effect of making the association between *LC* and *LCEQ one-to-one* since it asserts that a key class of *LCEQ* is composed of the same attribute classes as the key class of *LC* that has been associated with the link class. The attribute class *eqtype* in *EQUIV* records which of the four types of equivalence is being asserted, as previously discussed.

Domains (Data Typing)

In the ELKA method, an attribute is only permitted to assume a value from a specific set of values, referred to as its *domain*.[12] A domain is implicitly associated with each attribute class. Associated with each domain and, in turn, with the attributes defined by that domain, is the set of operations that may be applied to its values. These operations may be predicates, yielding truth values, or they may yield values in the same or some other domain.

An approach to encorporating domain declarations into the ELKA method showing some promise utilizes the relatively new concept of *abstract data type*. Abstract data types are used to characterize domains in terms of abstract properties in a requirements driven rather than available-representation driven fashion. Furthermore, the concept of *generic* data type gives this approach the capability of characterizing the abstract properties of many domains at once by means of parameterization. The abstract data type approach also provides the foundation for defining calculated or derived attribute values in terms of inter-type abstract operations.

In formulating a domain definition approach for the ELKA method, the domain and domain-like facilities of some common database definition methods have been reviewed to determine if they would be adequate or adaptable for information modeling purposes. Without exception, these methods view domains as being of two or three primitive types. Attributes

[12] This section is based, in large part, on Tardo, J.J., *Domains and Data Types In The ERA Information Model*, Hughes Aircraft Company, 1979. At an eary stage in its development ELKA was referred to by the name ERA.

(or their counterparts) are each associated with a primitive "domain," such as **integer** or **character** string, and usually supplied with some additional constraints to restrict range, field length (number of characters or digits), *etc.* Primitive "domains" are also equipped with a repertoire of operations, for example, to add integers or to test substrings. In essence, the concept of domain in these systems is almost identical to the conventional programming language concept of data type, with limited additional information, such as "units," "description," some constraints, *etc.*

There are some serious problems with applying this simple approach in the ELKA method. One such problem is that the primitive "domain" concept often admits semantically meaningless calculations. For example, an attribute whose values are supposed to be telephone numbers might be specified as having the domain integer, restricted to exactly seven (or ten) digits. However, this implies that telephone numbers may be added and subtracted. Defining them as strings of characters (restricted to digits) is little better, as characters usually come predefined with some lexical ordering scheme, which implies that two telephone numbers may be compared to see which one is "larger." It makes sense to ask if two telephone numbers are the same, but what possible distinction can be attached to *one being "larger" than another*? Clearly, restricting the range of values alone is unsatisfactory: to retain semantic validity the set of allowed operations must be restricted as well, or possibly even extended.

Another semantic breach can occur when values are compared, or otherwise operated upon across domains. The "units" mechanisms might prevent a comparison between, say, character strings representing telephone numbers and character strings representing purchase requisition numbers, but in general there are so many exceptions to this rule that units distinctions tend to be relegated to a status little better than optional comments. What units are attached to names or dates, for example? Can one multiply the *quantity* of a given part by the *weight* of that part, since quantity is "dimensionless"? Stating all of the rules, special cases, and exceptions is laborious, if not error prone, and hence is just not done.

The question of significant *vs* non-significant values should also be addressed at this point. A significant value is one that allows additional information to be inferred through its representation. In many databases, "significance" conventions are employed, for example incorporating the date into numbering schemes in order to ensure uniqueness or traceability. By exploding such a number into component sub-fields one may infer when it was created or possibly who created it. But such information is more appropriately maintained in distinct

attributes: The ELKA method, in fact, assumes that each attribute represents an atomic element of information and cannot be "exploded." While "significance" schemes are often important in the implementation of efficient information systems, they tend to obscure the information they encode. The purpose of information modeling is to highlight the information managed and utilized by such systems — it is necessary that representation and implementation efficiency issues not be allowed to obscure that information.

Looking from a different perspective, the technique of defining domains by restriction of primitive "domains" amounts to a *bottom-up* approach, attempting to mold solutions to fit the problems at hand rather than *vice versa*. More suitable for the ELKA method is a *top-down* approach, whereby one identifies the information requirements imposed on the domain in terms of the operations required of it. Representation issues can be deferred until an implementation[13] of the model is constructed so that, in general, different implementations might use different representations but still be described by the same model.

A *data abstraction* is a group of related functions or operations that act upon particular classes of objects under the constraint that the behavior of those objects can be observed only by applications of the given operations.[14] An *abstract data type specification* (or simply abstract data type) is a representation-independent definition of the operations that collectively define a particular class of objects.[15] Note that data type specifications in general interact, so that the operations described may involve arguments and values drawn from other classes of objects (data types).

In the sense used above, then, "abstract" should be understood as meaning "independent of representation." This is consistent with the programming language notion of *information hiding*[16] in which the operations of a data type are said to be collected into a *multiprocedure module*,[17] so that the only information that other modules may obtain about the values maintained internally is via the external interface operations. Hence the interface between modules is simplified and the "encapsulation" effect further prevents programmers from exploiting representation details that may change later. It also permits parallel development

[13] That is, an extension of the model — an information system based on a model.

[14] Liskov, B. and Zilles, S.: "An Introduction to Formal Specifications of Data Abstractions," in *Current Trends in Programming Methodology I* (R. T. Yeh, Editor); Prentice-Hall; 1977; pp. 1-32.

[15] Guttag, J., Horowitz, E., and Musser, D.: "The Design of Data Type Specifications," in *Current Trends in Programming Methodology IV* (R. T. Yeh, Editor); Prentice-Hall; 1978; pp.60-79

[16] Parnas, D.: "A technique for specification of Software Modules, with Examples," *CACM*; 15; 1972; pp. 330-336.

[17] See Liskov and Zilles, 1977.

of the multiprocedure modules themselves with the user modules.[15]

In the ELKA method, the concept of domain is identical with that of abstract data type. As an illustration, consider an entity with an attribute **Date-Received**, where this attribute has the domain date. (Domain [data type] names will be underlined.) It is not of particular interest how dates are represented — Julian date, manufacturing date, calendar date — but rather how dates may be *used*. It makes no sense to add dates directly, although we may wish to increment a date by some specific interval of time; however, it is necessary to be able to compare two dates. Similarly, **Vendor-Telephone-Number** values may be considered as being drawn from the domain telephone-number; applicable operations would include a predicate to compare two telephone-number values to determine if they are the same. The potential for performing arithmetic on telephone-numbers is undesirable. However, an interesting although un-explored opportunity presented by this approach is the ability to create composite "attibutes" through complex domains. In this sense, one might declare *EMPLOYEE*[...*phone_number*, *area_code*, *exchange*, *extension*...] where *EMPLOYEE.phone_number* is derived through the (domain-specified) concatination of the three attributes *EMPLOYEE.area_code*, *EMPLOYEE.exchange* and *EMPLOYEE.extension*.

The use of abstract data types to specify domains permits the ELKA modeler discretion as to what level of abstraction is most appropriate for attribute descriptions. It would also allow constraints that deal with purely domain-definition questions to be included in a "data sub-model." This would promote independent development of data-typing standards that could be applied across many models. The proposed approach to domains informally described here would capture semantically meaningful domain definitions in a way compatible with and complementary to the ELKA information modeling method. This approach utilizes and applies known research results in the area of abstract data type specifications, including the notion of generic data types. The rationale behind the selection of the abstract data type approach over the more primitive "data type" approaches currently in vogue in algorithmic programming languages is based principally on semantic issues. Furthermore, the abstract data type approach makes clear just which operations may be combined to generate calculated or derived attribute values without violating semantic integrity.

Further study and clarification are, however, required. Chief among outstanding questions is how abstract data type definitions should be wedded with the current method. It

[15] This approach is known more popularly today as *object-oriented programming*. When this work was originally done (in 1978–79) that term had not yet become current — the basic concepts, however, were as good then as they are now.

is unclear whether it would be better to provide a separate data sub-model integrated with the current modeling method, or to consider other graphic data modeling techniques, such as polyadic graphs. A comprehensive theory (algebra) for expressing and combining generic data type definitions needs to be developed. Also, the extent to which formalizing and testing of domain definitions should be carried out needs to be resolved. Finally, the data type definition approach will need to be integrated into the ELKA metamodel.

Paths

Link classes are considered to be directional in that they are said to each have an independent and a dependent end. When an entity class is located at the dependent end of a link class, it is said to contain members each of whose existence is dependent (through members of the link class) on members of the entity class located at the independent end of the link class. In fact, the effect of a link class is to establish a necessary existential dependence of members of one entity class on members of another entity class.[19]

A link class represents a collection of singular associations that pair entities through *identical* semantics. When a link class L is shown connecting an entity class A with an entity class B such that B is dependent (in L) on A, the interpretation is that the existence of each member of the class B is dependent on the existence of a member of the class A through a member of the class L. This may be thought of as defining a function L that maps members of B uniquely to members of A:

$$L(B) - A$$

This means that, from knowledge of some $b_i \in B$ we can infer the existence of an $a_i \in A$ through the semantics of L. Further, in the more general case of entity classes A, B and C, C dependent on B which is dependent on A through L_1 and L_2 respectively, we can infer from any $c_i \in C$ the existence of an $a_i \in A$ by way of an intervening $b_k \in B$. These inferences, simple and extended, are called inference paths. They lead to one of the unique capabilities of the ELKA modeling technique — the ability to recognize and express potentially contradictory semantics of association within a model.

An inference path represents a case of extended semantics within a model. By following these paths it is sometimes possible to "navigate" from one entity class to another through

[19] In some cases, a link class may associate members of an entity class with members of that same entity class.

more than one path,[20] thus identifying an ambiguity within the model. Take the case in which the entity class C is dependent on the entity classes B and A, and B is dependent on A. Beginning with a member of C, it is possible to infer the existence of a member of A directly *and* by first inferring the existence of a member of B. However, the semantics of $L_1(C) - A$ and $[L_2(C) - B$ and $L_3(B) - A]$ must be reconciled. If $c_i \in C \Rightarrow a_j \in A$ through L_1, is it necessary that $c_i \Rightarrow b_k \in B \Rightarrow a_j$ (through L_2 and L_3)? Or is it required that $c_i \Rightarrow b_k \in B \Rightarrow a_l \in A$ where $a_j \neq a_l$? Or does it matter at all? In practice, reconciling these semantic ambiguities is of great importance in contributing to the overall value of a model.

A simple notation serves as a shorthand for expressing these inference paths:

$$C - (L_2)B - (L_3)A \text{ and } C - (L_1)A$$

where C, B and A are entity classes and L_1, L_2 and L_3 are link classes. In those cases where no ambiguity of expression results, the link class names (appearing in parentheses) may be omitted.[21] The disjointness of paths is indicated with a not equal symbol:

$$C - B - A \neq C - A.$$

Similarly, comparability of semantics is expressed with an equivalence symbol:

$$C - B - A \equiv C - A.$$

Two cases, presenting slightly different problems, must be resolved: forks and loops. A fork is a situation like those illustrated so far — two or more paths lead from one entity class to another entity class. A loop is a situation in which an inference path leads from an entity class back to itself.

Although the issues to be addressed are relatively straight forward, insubstantial progress has been made in addressing them to date. There are three major issues:

a. given a model, what are the potentially conflicting inference paths through it;

---

[20] Note that we are discussing paths *through a model* and not paths through an "information base." Hence these paths occur at the class level, they lead from one *class* to another *class* of entities. At the member level (in an "information base," for example), such paths would actually be instantiated; they would lead from one *entity* to another *entity*. This terminology is at variance with other usage within the ELKA technique, where model elements are consistently referred to as classes having membership. However, here it is the path through a model that is of interest; if it were to be of interest to discuss the "instantiations" of them, we would refer to a *walk* as leading along a path from one entity to another entity.

[21] ...as has been seen throughout this report.

b. what is the minimum set of inference path conflicts that need to be resolved, allowing all others to be derived from those resolved;

c. given the resolution of some number of inference path conflicts, can they be proved not to be contradictory and how?

The problem has been approached from a set-theoretic perspective, but the mathematics have proved to be quite difficult. Another approach is to consider the model as a semantic net. In this way, link classes can be considered as rule sets, mapping members of dependent nodes onto members of independent nodes. Semantically comparable inference paths may be treated as co-implicants. This approach shows a great deal of promise being close to the modeling theory and apparently relatively straight forward.

The ELKA metamodel (and hence the rules being presented here) can be extended to represent inference paths in the following manner. An inference path (*IP*) is defined to be a continuous[22] chain of inferences (*INF*). A *LC* may allow a number of such inferences (*ie.* the *LC* might appear in several *IP*s). An *IP* is said to have a *LC* at its "head" and also at its "tail."

Hence,

$$IP[pathid, h_dep, h_indep, h_lcid, t_dep, t_indep, t_lcid] \tag{38}$$

$$\forall IP[pathid]_i : \\ (\neg\exists IP[pathid]_j \mid (IP[pathid]_i = IP[pathid]_j) \wedge (i \neq j)) \tag{39}$$

$$IP[h_dep, h_indep, h_lcid] \subseteq LC[dep, indep, lcid] \tag{40}$$

$$IP[t_dep, t_indep, t_lcid] \subseteq LC[dep, indep, lcid] \tag{41}$$

And,

$$Inf[path, dep, indep, lcid] \tag{42}$$

$$\forall Inf[path, dep, indep, lcid]_i : \\ (\neg\exists Inf[path, dep, indep, lcid]_j \mid \\ (Inf[path, dep, indep, lcid]_j = Inf[path, dep, indep, lcid]_i) \\ \wedge (i \neq j)) \tag{43}$$

$$Inf[path] \subseteq IP[pathid] \tag{44}$$

[22] There is currently no mechanism for enforcing this contiguity.

$$Inf[dep.indep.leid] \subseteq LC[dep.indep.icid] \tag{45}$$

Further.

$$\exists Inf - IP \mid Inf - IP - (head)LC \equiv Inf - LC \tag{46}$$

$$\exists Inf - IP \mid Inf - IP - (tail)LC \equiv Inf - LC \tag{47}$$

That is, every path (IP) has to contain an inference (Inf) that is its head and an inference that is its tail.

Rules that compare inference paths must be declared in order to resolve inference semantics within the model. An implication rule (IR) compares two inference paths. Such a rule states either that the independent entity found at the "head" of a walk along one path must be the same entity that is found at the (independent end of the) "head" of the other path when starting with the same dependent entity at the "tail" of the walks along both paths (*ie*, the paths are co-implicants); or that the two resultant (independent) entities may not be the same.[23] An implication predicate ($Pred$) allows implication rules to be qualified. So, we have:

$$IR[left_path.right_path.rule] \tag{48}$$

$$\begin{aligned} &\forall IR[left_path.right_path]_i : \\ &\quad (\neg\exists IR[left_path.right_path]_j \mid \\ &\quad (IR[left_path.right_path]_i = IR[left_path.right_path]_j) \\ &\quad \wedge (i \neq j)) \end{aligned} \tag{49}$$

$$IR[left_path] \subseteq IP[pathid] \tag{50}$$

$$IR[right_path] \subseteq IP[pathid] \tag{51}$$

$$Pred[left_path.right_path.pred] \tag{52}$$

$$\begin{aligned} &\forall Pred[left_path.right_path]_j : \\ &\quad (\neg\exists Pred[left_path.right_path]_i = Pred[left_path.right_path]_j) \\ &\quad \wedge (i \neq j)) \end{aligned} \tag{53}$$

$$Pred[left_path.right_path] \subseteq IR[left_path.right_path] \tag{54}$$

$$IR - (left)IP \neq IR - (\text{right})IP \tag{55}$$

Rule 55 states that an implication rule cannot compare an inference path to itself.

[23] The "default" case, then, is that it doesn't make any difference. That is, implication rules will only be recorded when one of these two cases holds.

Loops And Cycles

A link class can be thought of as describing a function that maps the members of its dependent entity class uniquely onto the membership of its independent entity class. Hence, we might express the case of an entity class B dependent on an entity class A through the link class L as

$$L(B) - A.$$

Further, we may state that

$$\forall b \in B \text{ and}$$

$$\forall l \in L \text{ (in which B is dependent)},$$

$$L(B) \Rightarrow a \in A.$$

The converse is not true. That is, each member of B must map uniquely onto a member of A but members of A may map onto several members of B but need not map to B at all.

A loop is a situation within an ELKA model in which an entity class is dependent upon itself. The length of a loop is the number of link classes that intervene between the entity class and itself. For example, we may model a company's organizational structure by noting that the organization is made up of departments, which fund the activities of employees. In fact, each department may fund several employees and each employee must be funded by exactly one department (in this simple example). Hence, a dependency is observed between *EMPLOYEE* and *DEPARTMENT* such that each member of *EMPLOYEE* (that is, each *employee*) must is existentially dependent on a particular member of *DEPARTMENT* (that is, a *department*). Further, suppose that we find that each department must be managed by exactly on employee but that employees are sometimes assigned to manage more that one department. A dependency is then formed between *DEPARTMENT* and *EMPLOYEE* such that each *department* is existentially dependent on exactly one *employee* (*ie*, its manager).

The model may now be expressed as

$$funds(EMPLOYEE) - DEPARTMENT \text{ and}$$

$$manages(DEPARTMENT) - EMPLOYEE.$$

This illustrates that a *department* must be managed by exactly one *employee* who must be funded by exactly one *department*. Two problems face us. First, is it allowed for the manager

of a department to be funded by that department? And, second, if not, how is it possible for us to represent top management — what department is available to fund them? The first problem is an inference path problem and relates to the potential differences between the semantics of the *DEPARTMENT* entity class and the semantics of association embodied in the link classes *funds* and *manages*. However, the second problem is a subcase of a more general problem with rooted networks. The root of a rooted network necessarily has no superordinate node. Since the structure we wish to represent is basically a rooted network (of a sort) we are forced to the conclusion that the "ultimate" manager must have no superordinate. But the modeling technique does not support this — the model says that each *employee* (manager or not) must be funded by a *department* that is managed by some *employee*. Hence, each *employee* must (through *DEPARTMENT*) have a superordinate (managing) *employee*. A common resolution of this is to say that the "ultimate" manager manages him or her self. This translates to mean that (at least) the "top" department is managed by an employee who is also funded by that department.

This example illustrates a loop of length two. A loop of length one is called a cycle. In essence, a cycle says that each member of the involved entity class must be dependent on *some member of the same entity class*:

$$L(A) - A \text{ or}$$

$$a_i \in A \Rightarrow a_j \in A.$$

And here we see the rooted network problem in its full glory. Each member must be dependent on some member, which must, in turn, be dependent on some member. . . and so forth. Resolution of this situation can only be achieved by allowing some members of the entity class to be dependent on themselves. These are the roots of the rooted network.

Actually, cycles among members of an entity class represent a special subcase of networks called trees. A tree is a rooted network in which each node may have many subordinates buy only one superordinate. In the usual notation of trees, a node with no superordinate is called the root of the tree and nodes with no subordinates are called the leaves of the tree. Within ELKA models, an entity that is a part of a cycle and has itself as its superordinate (that is, that is dependent on itself) is called the root of the tree, while members of that entity class that have no subordinates (that is, that have no members of the entity class dependent on them) are called the leaves of the tree. Cycles are commonly used in ELKA modeling to represent trees (or hierarchies), and this is always ultimately wrong. In fact, at the class level, a cycle does not so much represent a tree as it does a forest.

Take, for example, the case of an employee who is said to supervise some number of other employees. We know that all employees must be supervised by exactly one employee and that (because of the rooted network problem) the "senior most" employee will be considered to supervise him or her self. However, the use of the cycle

$$supervises(EMPLOYEE) - EMPLOYEE$$

does not necessarily represent a tree structure. It represents a family of similar tree structures. Each employee must be supervised by someone and some employees are supervised by themselves. Hence, we may imagine clusters of employees, each beginning with some (root) independent supervisor and leading down through some number of layers to some number of (leaf) non-supervisory employees. The problem of how we restrict forests to tress is as yet unresolved.

One final observation on trees and forests is that entangled forests make rooted networks. Hence, we might choose to represent a bill of materials structure (generally viewed as a rooted network) as a pair of cycles involving the entity class *PART*:

$$used_in(PART) - PART \text{ and}$$

$$contains(PART) - PART.$$

The first cycle, $used_in(PART)$, represents the forest of parts whose roots are indivisible parts — "components" — and whose leaves are systems in which the components appear. The second cycle, $contains(PART)$, represents the forest of parts whose roots are systems and whose leaves are the components that make up those systems. Traversing the first cycle results in all of the uses that are made of a component. Traversing the second cycle results in all of the assemblies and components that make up a system. The entanglement of these two forests leads to the network structure with which we are familiar when dealing with bills of materials.

5 Bibliography

Here we have listed a very few of the writings that influenced us in the development of ELKA and a few of those that have been produced in the meantime having relevance to our efforts.

Motivating Literature

Bachman, C.W.: "Data Structure Diagrams.": *Data Base*, No. 1 Vol. 2, Summer 1969.

Backman, C.W.: "The Role Concept in Data Models.": *International Conference on Very Large Data Bases*, 1977: pp. 464–476.

Biller, H. and Neuhold, E.J.: "Concepts for Conceptual Schema.": *Architecture and Models in Data Base Management Systems*, North-Holland, 1977: pp. 1–30.

Chen, P.P.: "The Entity-Relationship Model — Toward a Unified View of Data.": *ACM Trans. on Database Systems*, Vol. 1, No. 1, March 1976: pp. 9–36.

Lockemann, P.C. and Newhold, E.J. (editors): *Systems for Large Data Bases:* North-Holland, 1976.

Nijssen, G.M. (editor): *Architecture and Models in Data Base Management Systems:* North-Holland, 1977.

Nijssen, G.M.: "Modeling in Data Base Management Systems.": *Proceedings, Euro IFIP*, North-Holland, 1979. (Invited paper.)

Senko, M.E.: "Conceptual Schemas, Abstract Data Structures, Enterprise Descriptions.": *International Computer Symposium, 1977*, North-Holland, 1977: pp. 85–102.

Tardo, J.J.: *Domains and Data Types In The ERA Information Model:* Hughes Aircraft Company: 1979. (At an early stage of development, ELKA was called ERA.)

van Griethuysen, J.J. (ed.): *Terminology For the Conceptual Schema And the Information Base:* ISO/TC97/SC5-N695: ISO Report, 1982.

Wilson, M.L.: "A requirements and design aid for relational data bases": *Proc. 5th Int'l Conf. Software Engineering*, IEEE, 1981: pp. 283–293.

Other Relevant Literature

Brodie, M.L., Mylopoulos, J. and Schmidt, J.W. (editors): *On Conceptual Modeling:* Springer-Verlag: 1984.

Lohman, G.M., Stoltzfus, J.C., Benson, A.N., Martin, M.D., and Cardinas, A.F.: "Remotely-Sensed Geophysical Databases: Experience and Implications For Generalized DBMS": *SIGMOD Record Volume 13, Number 4*, Order No. 472830: ACM: 1983: pp. 146–160.

Ramey, T.L., Brown, R.R., Melkanoff, M.A., and Rodriguez-Ortiz, G.: *The ELKA Information Model:* Hughes Aircraft Company: 1982.

Ruoff, K.L.: "Practical Application of IDEF1 as a Database Development Tool": *Proceedings of the International Conference on Data Engineering.* IEEE: April 1984: pp. 408–415.

Tardo, J.J.: *The Design, Specification and Implementation Of OBJ-T: A Language for Writing and Testing Abstract Algebraic Program Specifications:* Doctoral dissertation: University of California, Los Angeles: 1980.

Thompson, P.: *Natural Language Analysis, Information Modeling, and Database Engineering:* NLIMDE5: Control Data Corporation: 1987.

van Griethuysen, J.J. and King, N.H. (editors): *Assessment Guidelines for Conceptual Schema Language Proposals:* ISO/TC97/SC21/WG5-3: ISO Report: August 1985.

Wilson, M.L.: *Information Automat:* TR 101: IA Systems, 112 Winsted Court: San José, CA: 1986.

DESCRIPTION OF A SEMANTIC DATA ENGINEERING AND PRODUCT DATA CONFIGURATION CONTROL SYSTEM ENVIRONMENT

E.I. (Sam) Nusinow

Knowledge Base Engineering Enterprises

Contributed by E.I. (Sam) Nusinow

INTRODUCTION

Effective product data configuration control has become an increasingly critical aspect of evolving integrated information systems technology within engineering environments. This paper discusses the need for a Semantic Data Engineering Tool (SDET) that can be used to create and maintain a Common Data Knowledge Base (CDKB) for use by a Product Data Configuration Control System (PDCCS). Together, this tool set would be used to collect, maintain, and control semantic data and operational control rules that are related to engineering processes operating within distributed heterogenous computer system environments.

The remainder of this document focuses on:

- the problems that are prevalent within engineering environments today.
- the technical objectives of a SDET, a CDKB, and a PDCCS.
- a technical approach to evolve SDET, CDKB, and PDCCS technology.
- the use of a SDET, a CDKB, and a PDCCS on a typical engineering project.

CURRENT PROBLEMS WITHIN ENGINEERING ENVIRONMENTS

Media Inconsistency

Today, many engineering companies manage technical product definition and support data (hereafter referred to as product data) that resides on both paper and electronic media. Within most of these companies much of the product data resides on paper in the form of drawings, engineering specifications, change orders, problem reports, quality assurance reports, etc. Currently, there is a thrust within industry to automate engineering processes to improve productivity. As a result, product data is in a state of transition from paper which is handled manually by people to electronic media which is handled in an automated fashion using computer systems.

Automated Product Data Configuration Control Inadequacy

Product data configuration control functions are used to identify, control changes to and maintain the integrity and traceability of a product configuration throughout its life cycle. There are few automated systems in place today that provide the functionality that is required to adequately perform product data configuration control in the distributed heterogeneous computer system/data base management system environment. The few systems that are available, fail to address the complex data object structures, life cycles,

interdependency relationships, etc., that are prevalent in this type environment. Most of the automated product data configuration control systems that have evolved to date, have been built around general purpose data base management systems (e.g., Codasyl and Relational) that pose problems in the following areas:

- o Product data specification
- o Complex consistency constraint checking
- o Complex aggregate object manipulation
- o Relationship type specification

Technological advances in automated product data configuration control are taking place, but at a very slow rate and with many important questions still unanswered.

Application Non-Integration

Product data configuration control procedures vary between the organizations of a company. These organizations tend to follow their own procedures and utilize various non-integrated applications. This causes problems such as: duplication of data, ineffective data communications and increased data utilization errors, etc.

For the most part, the applications being used tend to be designed to address a specific set of functions, they are developed independent of

one another, and they do not share the product data that they utilize. Since the applications are non-integrated, the product data that they use is duplicated in more than one data base, is inconsistent and non-synchronized, and is difficult to transfer from one organization to another due to format and semantic differences. Another major stumbling block is that these applications do not effectively manage and control the evolution of product data. This is because there is no global knowledge base that contains control information about the product data.

Engineering Process/Product Data Life Cycle Analysis Complexity

Product data evolves through stages of planning, design, development, implementation and maintenance. These stages represent the life cycle of product data. A product data life cycle has many complex interdependencies that affect its evolution. For example, a product engineering change cannot be released until all related data objects (e.g., geometry data, analysis data, test data, specifications, etc.) have reached the state of "Approved." Within a large engineering environment there can be many related data objects, each having their own life cycle. It is the product data interdependencies such as this, that create the need for an in depth analysis of the engineering process and product data life cycle evolution.

Currently, methods used to conduct an engineering process/product data life cycle analysis are time consuming, and repetitive activities.

During the analysis effort, interviews are conducted with engineering experts who provide system engineering analysts with detailed product data definition/flow/processing information. The analysts then prepare models (i.e., function, information, process, simulation, etc.) that graphically depict the engineering environment and process, and; the rules, life cycles states and interdependency relationships that are associated with the product data. The models are either manually prepared or developed using Computer Aided Software Engineering (CASE) tools. Once they are prepared, the engineering experts review the models to identify mistakes and deficiencies. These activities are repeated until the experts are satisfied that the models are accurate. The information contained in the models is then manually translated into requirements and design specifications which again must be reviewed by the experts prior to detailed system design and implementation.

Semantic Data Engineering Tool Deficiencies

Most of the analysis tools that are available in industry today primarily support the design and development of software. What is needed, are semantic data engineering tools that can support the design, development, and implementation of product data configuration control system technology. Of the CASE tools that are available, very few, if any, collect, analyze, define, verify, store and maintain the semantic data and operational control rules that are required to design, develop and implement product data configuration control

systems.

Within the context of this particular discussion, it is important to note that there also needs to be a capability for interpreting the meaning of the semantic data and operational control rules so that they can be graphically displayed in the form of models. Many of the currently available software engineering tools process model element input in a syntactical fashion, i.e., they do not interpret the meaning of model elements. For example, an information model contains syntax such as boxes, arcs and circles, which semantically represent entities, relationships between entities (e.g., a-part-of, an-instance-of, a-kind-of, connected-to, is-used-for, sends-to, receives-from, etc.) and entity relationship cardinality. It is these types of semantic representations that a semantic data engineering tool should be able to interpret.

Of equal importance, is the need to capture the semantics that are represented on process models, i.e., the activities that must be performed in a given sequence, the life cycle states that the product data evolves through, the constraints that must be satisfied before product data can change state and the mechanisms that are required to perform the activities. This information is required so that product data can effectively be controlled during its life cycle.

TECHNICAL OBJECTIVES

A major technical objective of the Knowledge Base Integrated Information Systems Engineering Project should be to establish base line functional requirements for a Semantic Data Engineering Tool (SDET), a Common Data Knowledge Base (CDKB) and a Product Data Configuration Control System (PDCCS) that are intended for use within distributed heterogeneous engineering environments.

The primary purpose of the SDET should be to support system analysis/ engineering activities. It should provide capabilities to collect, analyze, define, verify, store, maintain, display and transmit the semantics and operational control rules associated with product data. Semantic data and operational control rules refer to data objects, data object attributes, data object structures, constraints, default/legal values, edit rules, legal operations, relationship types, interdependency relationships, process activities, life cycle states, state change approval rules, authorized user groups, notification/distribution rules and user views that are required to support the PDCCS.

System engineering analysts should use the SDET to maintain a local CDKB that contains in-process semantic data and operational control rules. The SDET should be able to send/receive subsets of the semantic data and operational control rules to and from a global CDKB for approval and release purposes. Once in the global CDKB, released semantic data and operational control rules could be used by the PDCCS

to perform product data life cycle control functions. In order to accomplish this, standard data exchange formats will have to be established.

The PDCCS should control product data life cycles and communicate data object types between several engineering processes operating in a distributed heterogeneous computer system/data base management system environment. It should provide engineering operational capabilities that enable users to identify, access/store, maintain and control product configuration item structures and related data objects. The PDCCS should utilize the semantic data and operational control rules contained in the global CDKB to enable users to relate data objects, request life cycle state changes, generate messages, approve/reject life cycle state change requests, etc.

The PDCCS should also provide capabilities that enable system engineering analysts to check-out subsets of semantic data and operational control rules contained in the released global CDKB for local SDET maintenance activities. It should allow users to determine the status of local in-process CDKB changes, approve issued CDKB changes, check-in approved subset versions of the CDKB, etc., in the same manner as it does for product data.

The PDCCS should track the life cycle state changes associated with each data object under its control and maintain an audit trail of the process activities that require completion before a data object can change state. It should provide a status of in-process changes with

visibility over all projects and job assignments. The PDCCS should automatically notify specific users when process activities are either initiated or completed.

TECHNICAL APPROACH

This section presents an approach for establishing base line functional requirements for the SDET, CDKB and PDCCS. Very little attention has been given to the development of a system engineering environment that can support the analysis and specification activities required to create and maintain a common data knowledge base. In order to determine the semantic data and operational control rule entity types that are required, the functionality to be provided by a SDET and a PDCCS must be identified.

Conduct Requirements Analysis

An analysis should be conducted to establish base line functional requirements for the SDET and the PDCCS. The requirements should be based on the needs of evolving government and commercial programs. The requirements should be used to establish research and development plans for production versions of the SDET, CDKB and the PDCCS.

The requirements analysis effort should focus on the review of requirements that have already been documented by Air Force programs

(e.g., the Integrated Information Support System, the Integrated Design Support System, and the Engineering Information System) and similar commercial programs (e.g., Honeywell Incorporated - Engineering Data Management System, and Ford Aerospace & Communications Corporation - Computer Aided Configuration Management System). These programs are currently addressing the problems associated with managing and controlling product data in a distributed heterogeneous computer system/data base management system environment.

Detailed interviews should be conducted with selected groups of personnel that are involved in these programs to:

- o evaluate a base line set of system engineering and product data configuration control requirements that are relevant to the design, development and maintenance of the SDET, CDKB, and PDCCS.

- o identify functions that should be provided by the SDET and the PDCCS.

- o identify the semantics associated with data objects, their life cycle states, and the process activities that control their evolution.

- o identify the data object types that should be maintained in the CDKB.

- o scope, categorize and prioritize requirements for the SDET, CDKB and the PDCCS.

Evolving object oriented database management systems should be examined to determine if the technology is applicable to the development of the SDET, CDKB and the PDCCS. Systems such as these are designed to handle the complex data structures and relationships that would be required by the SDET, CDKB and the PDCCS.

Product data exchange standards currently under development by the National Bureau of Standards (NBS) and other organizations within the government should be examined to determine if they can satisfy SDET/PDCCS data (i.e., semantic data and operational control rule) exchange requirements. Product data exchange standards (e.g., PDES) are designed to provide a neutral exchange medium capable of completely representing product data for the purposes of exchanging such data between dissimilar CAE/CAD/CAM systems.

Develop Models

Function, Information, and User Interface modeling techniques should be utilized to produce high level models that specify "To Be" SDET, CDKB and PDCCS requirements. The models should define the functions, identify the entities, and illustrate the screens and menus that are to be supported by the SDET, local/global CDKB and PDCCS. The function models should define the inputs, controls, functions, mechanisms, and

outputs that are to be supported by the SDET and PDCCS. The information models should identify the entities, relationships, attributes and constraints that are to be supported by the local/global CDKB. The User Interface models should illustrate SDET/PDCCS prototype menus, screens, user functions and most importantly, how they are presented to the user (dependent upon the user type and system function that is to be performed).

Establish Methodology Requirements

Requirements should be established for a methodology that allows SDET system engineering users to collect, analyze and define the semantic data and operational control rules that are to be maintained in the local/global CDKB. The methodology should be automated such that it enables users to interactively maintain the local and global CDKB. The methodology must be robust enough so that the data objects, data object relationships, data object life cycles, rules, facts, etc., that are associated with the engineering process can be described in detail. The methodology should govern the evolution of the data that is contained in the local/global CDKB, i.e., it should be loaded, tested, approved, released, controlled and maintained.

Develop SDET Prototype

Several companies and government organizations are currently developing

prototypes that perform PDCCS functionality. Presently, none of these prototypes support system analysis/engineering activities to collect, analyze, define, verify, store, maintain, display and transmit semantics and operational control rules as would an SDET.

An expert system based prototype should be developed to demonstrate SDET functionality. Rapid application development techniques should be utilized to build an experimental CDKB and user interface. The prototype development activity should focus on the creation of data entry/display screens, integrity constraint algorithms, local CDKB maintenance functions, etc., to demonstrate how semantic data and operational control rules can be collected, analyzed, defined, verified, stored, maintained, interpreted and displayed. The prototype should also demonstrate how semantic data and operational control rules can be transmitted between local and global CDKBs.

To determine the feasibility of the proposed approach, SDET prototype functionality should be developed utilizing commercially available expert system development tools. These tools should be used to develop design rules that support the semantic data/operational control rule specification methodology. The design rules would guide the user during semantic data and operational control rule acquisition and specification to insure data integrity and consistency of the CDKB. SDET users will require powerful, yet easy-to-use, user interface capabilities to facilitate the acqusition/specification process. "Mouse" activated graphics symbols (ICONs), multiple windows, pull down menus, desktop file management, etc., are such capabilities.

USING THE SDET/CDKB/PDCCS ON AN ENGINEERING PROJECT

This section presents a scenario of the project management and engineering activities that occur during the life cycle of an engineering project, from initial customer contact through project close-out. The scenario is provided to establish a general framework for describing how a SDET, a CDKB and a PDCCS could be applied within an engineering environment.

Before a project even begins, a base line project management plan is stored in the global CDKB. The plan is developed using the SDET and local CDKB. System engineering analysts identify engineering organization structures, engineering tasks, engineering data types, change control procedures (e.g., life cycle state change approval rules), authorization/access control rules, notification rules, project deliverables, etc. When the base line project management plan is completed, it is up-loaded from the local CDKB to the global CDKB so that it can be used during an engineering project.

During a preliminary phase, still before a project actually begins, an engineering company consultant works with a customer to determine the nature of a problem/need. The customer may be having difficulties modifying an existing product design or may wish to have a new product completely designed, analyzed, tested, etc., from scratch. It is the engineering consultant's responsibility to develop a high level

statement of the problem/need, a base line set of requirements that are to be satisfied by a proposed solution, and an agreed upon approach to solve the problem/need. This preliminary customer/engineering consultant interaction is referred to as the overview analysis phase.

During the overview analysis phase, the SDET is used to down-load a copy of the base line project management plan from the global CDKB to the local CDKB. The base line project management plan is used as an initial checklist to indicate what tasks will be required, what engineering tools will need to be used and what data object types will be produced to solve the customer problem/need. Selected items on the checklist are reviewed with the customer or against the RFP (Request For Proposal) to establish rough project cost/time estimates. Any items that have not been previously included on the checklist may be added at this time if required.

Next, the engineering consultant works with an engineering company management team to develop a proposal that includes a technical approach for solving the customer problem/need and a project plan that consists of the following items: task descriptions, deliverables, an organization structure chart, resources (i.e., manpower and equipment), a cost break-down chart and a schedule. The base line project management plan contained in the local CDKB is evaluated by the engineering project manager and the engineering consultant to determine if the best technical approach/plan to solve the problem/need has been chosen. At this point, the base line project management plan becomes a project "specific" management plan. Activities, resources and

deliverables are added to or changed in the project management plan contained in the local CDKB if it does not satisfy project requirements. Time/cost estimates are also updated to reflect the latest modifications to the plan. This process is repeated for each selected item of the plan until the local CDKB has been approved for project usage. The SDET report writer will then be used to print out the final version of the project management plan. The printed version is included in the proposal that is presented to the customer. It contains the following information:

- Technical Approach
- Project Tasks
- Project Deliverables
- Organization Structure
- Project Team Members
- Task Schedule
- Resources
- Cost Break-down

Upon receipt by the customer, the proposal is reviewed for approval or rejection. If approved, a contract is negotiated between the customer and the engineering company. During contract negotiations it is possible that the customer may wish to change the project management plan by either adding or eliminating some of the tasks and/or deliverables. Modifications such as these, will more likely than not, change the dollar figures that were specified in the cost break-down section of the proposal. When both parties are satisfied with the

outlined tasks, deliverables and associated costs, the contract is signed.

To ensure that proposed results are attained during a project, top level management commitment and integral involvement is essential. It is for this reason, that a project management team is organized. The project management team is comprised of representatives from various customer organizations and the engineering company. The project management team must concern itself with operational issues related to the planned strategy, i.e., making sure that tasks are performed on schedule and within budget, and that agreed upon deliverables are produced. The individuals that make up the team must participate in engineering project review meetings, must work together to produce project status reports, must monitor/regulate project costs and must coordinate/control resources to achieve planned project objectives. It is the teams responsibility to set qualitative and quantitative standards for processes and products developed during the project; evaluate the performance of the project; and adjust plans, policies, activities, resources, etc., accordingly.

The first event on the project agenda is to attend a project kick-off meeting. At this meeting, that the engineering project manager, consultant, lead engineers, engineers, technicians, and the customer project team are all assembled. The purpose of the kick-off meeting is to acquaint everyone involved in the project (i.e., the project team) with the technical approach, purpose and milestones outlined in the project management plan. Any modifications to the technical approach

that may be required are made during this meeting. The engineering project manager reviews the project management plan with the entire project team to determine if any changes are required in terms of personnel, resources (hardware/software), engineering process assignments, tasks, schedule, deliverables, project organization structure, etc. If so, the project management plan contained in the local CDKB is updated to reflect the actual project assignment changes that have been identified. When the project management plan is approved for release, it is up-loaded from the local CDKB to the global CDKB for use by the PDCCS.

Following the project kick-off meeting, tasks specified in the project management plan are initiated. The first major engineering activity is to conceptualize the design, analysis and/or test solution. Together, the engineering consultant and the project management team review the statement of problem/need and the base line set of requirements that were developed during the overview analysis. Depending upon the nature of the problem/need and the comprehensiveness of the requirements, further interaction between customer personnel and the engineering consultant and/or project manager may be required. If this is the case, then the engineering consultant and/or the project manager work with customer personnel to more accurately define the requirements and conceptualize a solution. It is possible that customer/engineering company interaction may occur many times during the life cycle of a project.

After a final solution to the customer problem/need has been

conceptualized, various engineering applications are utilized to actually develop the solution. To accomplish this, project engineers and technicians start by creating computer models, e.g., mechanical product design geometry. Once created, the design geometry must then be modified and/or reformatted for static/dynamic analysis and stress test processing.

Computer model design, analysis and test processes are of an iterative nature (i.e., may be performed many times for one design concept). During each process several alternate designs may be generated. In many cases, results from one computer run are used as input for subsequent computer runs. These results may then be used to make design modifications or used as input for other analysis and test runs. This process continues until the overall set of customer requirements are satisfied.

The PDCCS is used as a mechanism to aid in the engineering project management/engineering process (See Figure 1). It is used to identify, track, control and status information related to the product data that is generated by the engineering applications during the course of the project. Accurately tracking the status of design, analysis and test data may be the single most important aspect of engineering project management and control. Data status must be available and up-to-date from the time that the data is originally created, to the time when it is archived for subsequent delivery to the customer, and even beyond. First and foremost, it is essential that the project management team [i.e., project manager and lead engineer(s)] be aware of the identity

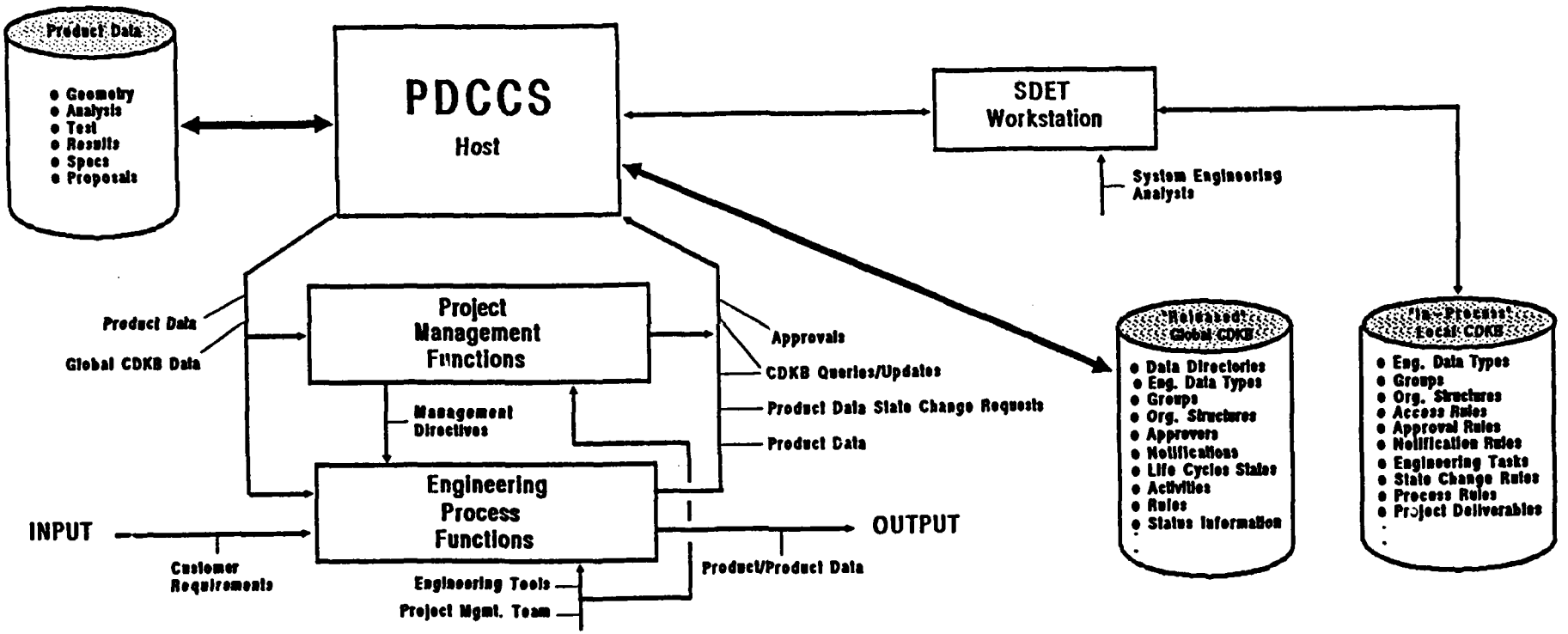


Figure 1 – Project Management/Engineering Process Control Model

and location of design, analysis and test data associated with given parts of a product structure. It is also important that the management team be kept abreast of the design, analysis and test results as they become available. It is their responsibility to evaluate results so that necessary modifications to designs and/or processes can be made and to communicate this information to the project team members and customer management.

Through the use of interactive PDCCS terminals, authorized project engineers and technicians access and update interrelated product data tracking/control information contained in the global CDKB. Using the PDCCS, an engineer requests state changes on product data that is ready to be reviewed and/or approved by the project management team for release to the customer. The project management team uses the PDCCS to respond to PDCCS generated notification/approval messages initiated by the engineer. Using the PDCCS, the project management team can obtain copies of controlled product data requiring review and approval. Once final results have been approved, the developed design concept can be released for final customer review and approval.

During project close-out, the project management team meets to decide on the format and content of a final report. At this time, the final report is outlined and tasks related to its development are identified. Upon completion, the final report is delivered to the customer and design models produced during the project are archived (i.e., off-loaded from the computer system hard disk to magnetic tape).

To summarize, it is clear that the SDET, CDKB and PDCCS all play important roles in the acquisition and utilization of semantic data and operational control rules to control the evolution of product data during an engineering project.

SUMMARY

Government and private industry experts have repeatedly voiced the need for a technology that can integrate product data of diverse engineering environments. Some companies have been actively involved in identifying requirements for this technology under government and commercially sponsored programs. It is evident that this technology is still in its infancy and that there is a need for additional functionality in this area. The successful development of the SDET, CDKB and PDCCS would satisfy the requirements to define, integrate, control and communicate product data between several engineering processes. Currently, there are no tools available in the marketplace that satisfy all these requirements. Potential post application of the SDET, CDKB, and PDCCS will be as a fully integrated tool for use by both government and industry and applied to information analysis, process analysis, semantic data definition, and configuration management and control.

NATURAL LANGUAGE INTERFACES FOR LARGE-SCALE INFORMATION PROCESSING

Lynette Hirschman

Unisys Defense Systems

Contributed by Lynette Hirschman

1. Introduction

This paper outlines the role and availability of natural language processing in managing large scale information processing applications, such as the large scale logistics applications characteristic of the Department of Transportation and the Department of Defense. The use of natural, as opposed to formal, languages is a key ingredient in facilitating human-machine communication, where "natural language" is taken to mean use of either spoken or written language, by the human and by the machine. Use of natural language raises the machine nearer the human level, rather than expecting the human to accommodate to the level of the machine. The vision in the discussion that follows is of *intelligent machine assistants* aiding the human decision maker in assimilating, organizing, storing, accessing, and integrating vast amounts of data. Without this assistance, it is clear that decision making will be labor-intensive, uneven in quality, slow, expensive, and probably insufficiently informed for optimal decision-making.

To date, natural language processing (NLP) has had very limited impact on interfaces to large-scale applications. This technology is only now maturing to the point where commercial applications are feasible; we expect major breakthroughs in the next five to ten years in: (1) providing custom systems for natural language access to distributed information systems; (2) capture of information in natural language (NL) form for building information systems; (3) providing limited "toolkits" which can be customized by end users for limited NL

interface applications; (4) combining NL techniques with other media: icons, menus, pointing devices, to provide succinct high-level interfaces; and (5) supporting communication via *spoken* language especially speech recognition, but also speech generation, in addition to written language. Realistically, a robust broad-coverage speech system is not likely to be available for at least six to eight years. However, there are a number of important applications of NL that are likely to be available within a five-year horizon. This report will describe the state of the art and identify some key applications with near-term payoff.

Although the obvious applications of NLP are to user interfaces, there are some other important contributions that this area will provide. For example, one of the major issues emphasized in the Executive Summary on Knowledge-Based Integrated Information Systems Engineering [11] is the ability to define a "global" semantics for distributed data bases. A study of how people talk about the data in such databases can provide the underpinnings to such a mapping. The semantics required to formulate an NL front-end may capture much of the global semantics required by such a system.

The specific issue of temporal processing is significant enough to merit special mention. In a dynamic situation, especially a decision support application, it is critical to be able to capture temporal information, retrieve it, reason about it, and talk about it. NL researchers have been among the chief groups addressing this issue, and their work can make major contributions to the processing of temporal relations. This report will discuss some of the ongoing work

in this area in section 6.1.

2. Dimensions of Natural Language Processing

This section describes various dimensions of natural language processing applications. These dimensions can be summarized as follows:

(1) **Directionality of communication:** human to computer (understanding) and computer to human (generation). In addition, although the focus is on natural language, we will discuss several "mixed" approaches, which rely on techniques other than NL techniques. In particular, section 3.2 discusses combining information retrieval techniques with natural language techniques; and section 5 discusses combining graphics, menu, and mouse interfaces with natural language.

(2) **Written vs. spoken language,** for both recognition and generation. Although spoken language generation has reached a reasonable level of sophistication, spoken language understanding remains a very difficult issue: speaker-independent continuous speech understanding with a non-trivial vocabulary size (2000+ words) is still a good five years away. One of the critical issues here will be the ability of a speech understanding system to adapt to the speech of different users. Current systems require extensive training for each new user; the training requirement for future systems should decrease, as research makes progress in ability to feedback information to the system, to provide constrained kinds of "learning".

(3) **Size of Domain**, ranging from a very narrow domain, covering very limited topics, to completely unconstrained discourse. To date, the major successes have been achieved by constraining the domain of discourse to a narrow, well-defined topic (e.g., newspaper stories about Cyrus Vance, database queries to a database with information on Fortune 500 companies, spoken text editing commands, etc.). A constrained domain limits vocabulary, it limits lexical ambiguity, and it limits the kind of things that can be talked about, making it possible to build a reasonably detailed domain model. As the domain expands or as multiple domains are covered, problems of domain modeling and lexical ambiguity can become severe. Although some systems attempt to cover an application area rather than a *domain* (e.g., *checking business correspondence for grammatical correctness* in EPISTLE[22, 14], coverage of broad domains significantly increases the complexity of the problem.

(4) **Complexity of utterance**, ranging from single sentence to complex multi-paragraph discourse. The difficulty grows in complexity as the task moves from single sentence utterances requiring relatively little in the way of discourse processing machinery, to complex discourse, which requires an elaborate set of modules for processing discourse relations, including machinery to carry along discourse context, as well as modules for reference resolution, temporal reasoning, and reconstruction of contextually implicit information. This applies to both recognition and generation.

Application types fall into three basic classes, which cut across the dimensions enumerated above. The three classes are:

a) **information access**
b) **information capture**
c) **mixed initiative interactions** (human-machine dialogue).

The first class of application, *information access*, is what usually comes to mind when discussing natural language interfaces to data/knowledge bases. This is the application area where commercial products are becoming available (e.g., Intellect, Q&A, Themis). The key requirement for information access is the use of natural language understanding to translate a natural language request into a DB/knowledge base query. The minimum requirement is ability to understand single sentence utterances. A more sophisticated interface would add the capability to understand a series of queries, including the ability to handle pronouns and ellipsis, but such a system requires substantially less sophistication than a system for information capture (see discussion below). Natural language generation can be used to enhance intelligibility of the answers in certain situations[15], as well as to offer feedback and/or explanations.

Information capture emphasizes the ability to understand paragraph and multi-paragraph input. This requires a substantially more sophisticated system than simple query-processing. Information capture via natural language has been an active area of research for over fifteen years, and is rapidly maturing. One indication of this is the *Message Understanding Conference*, to be held in San Diego June 10-12, 1987, sponsored by NOSC (Naval Ocean Systems

Command). The goal of the conference is to demonstrate and compare capabilities of various message processing systems, given a Navy domain of intelligence messages (RAINFORMS). Given the state of the technology, it seems likely that there will systems routinely capturing natural language text input in limited domains within the next five to seven years (e.g., medical reports[12], equipment failure reports[19, 7], intelligence messages[1], and bank telex messages[18].

The third area, mixed initiative or dialogue interaction, requires the most sophistication, since it is a more open-ended situation which requires that the machine adapt to a variety of user-responses for help or information. It requires an ability to handle connected discourse (as does the information capture application); it requires an ability to generate coherent explanations of its answers; and finally it requires an ability to provide "cooperative responses" to a range of users, including users who become increasingly sophisticated as they gain familiarity with the system. This in turn requires the ability to model the users of the system and to update these models over time.

3. Information Access via Natural Language

Accessing information (e.g., databases or knowledge bases) via natural language is the most widely known application of natural language, probably because it is the only one that has had some significant commercial distribution in the form of systems such as *Intellect*, *Q&A*, and *Themis*. In addition, there are several powerful research systems, including SRI's TEAM system[9], and BBN's IRUS system[34]. By allowing a user to write queries in English, rather

than in a formal query language, the information in the DB becomes more easily accessible to a wider community of users.

Accessing information via natural language raises a host of issues and implementation questions:

Portability

Can the system be ported to a different DB? To a different DBMS? To a different hardware/software environment?

Coverage

What portion of the language does it cover? Is this portion enough for users to stay inside it? Do users have to be trained to stay inside the language subset?

Reliability

What mistakes does the system make? Can it reliably recognize its own failure to understand? How does it help the user to "fix" a request that it hasn't understood? How does the system handle something outside the DB domain?

Intelligibility

How does the system feed the information back to the user? Does it offer an explanation? Does it request additional information in cases of ambiguity? Does it catch false assumptions on the part of the user?

System Integration

How does the system fit into the overall data-processing architecture? Does it require special hardware/software? Can it be integrated into other facilities (report generation tools, pointing devices, etc.)? Can the system be used to access distributed databases? knowledge bases?

3.1. Requirement for Access to Distributed Database Systems

The current systems provide only a very small portion of the capabilities outlined above. In addition, for complex decision-making applications, it is not clear that access to individual databases in isolation, by whatever means, addresses the information processing problem. What is really required is intelligent access to heterogeneous distributed data management systems, since in many applications, data may be distributed, for historical reasons, across a number of different systems; finding a solution will involve accessing different kinds of data from the different systems and aggregating the data (using some kind of inference or reasoning procedure) and finally displaying the answer(s) in an appropriate fashion. The ideal architecture would then be a powerful natural language capability interfacing to an Expert Distributed Database system, which would be able to retrieve data from different DB's and reason about it.

Since a natural language DB interface must define a mapping between the database semantics and the semantics of a given language, it is also natural to look to natural language processing techniques to aid in defining a "global"

semantic mapping. In particular, the kind of information that a natural language interface needs, namely the aggregation (part-whole) hierarchy and the specialization/generalization (is-a) hierarchy are two important methods of organizing information in defining database semantics. Natural language and DB technology need to interact in this area, to improve tools for providing a generic semantic description of databases that can be used both for global schema definition and NL interfaces.

3.2. Browsing through Large Databases

For some applications, what is required is not retrieval of a specific set of facts, but the ability to "browse" through the database. This is particularly important when 1) the user does not know exactly what information is needed, and 2) the volume of data is very large, so that a query might retrieve unmanageable amounts of data. This would be typical, for example, of search using key words, where the user tends to formulate retrieval requests based on feedback from the system. For example, if a retrieval request to a database containing equipment failure reports for all reports about *tapes* retrieves several thousand entries, the user will want to narrow the search to *bad magnetic tapes*, to retrieve only those reports of direct interest. Alternatively, if an initial request for *mag tape* retrieves nothing, the user might want to generalize the query to a search for *bad tapes*, or possibly may rephrase the query entirely, to retrieve the desired data.

To date, there has been interest in natural language techniques as applied to information retrieval problems, but few successful systems (even research systems) based on any kind of linguistic processing have been realized. One successful natural language front-end to an information retrieval system is the NLM CITE system [5], which provides a natural language interface to MEDLINE, a major medical bibliographic information retrieval system.

Even more experimental are the systems used to encode information for use as index terms in retrieval. The *RECONSIDER* system [32] offers an interesting approach based on the *structured text* found in *Current Medical Information and Technology (CMIT).* The techniques in RECONSIDER are not linguistic, but exploit the regularities in CMIT to create inverted files of co-occurring terms, used for retrieval of diseases associated with symptoms. Limited natural language techniques have also been applied to generation of subject indexes[33] and to chemical reaction databases[28].

Although augmenting information retrieval with natural language techniques has received some attention, it is still an under-explored area that could have significant near-term pay-off. For example, one strategy might be to use key word search to retrieval sentence units containing a particular word. followed by natural language processing, to determine whether the key word is in the appropriate context. By getting feedback from the user about which occurrences were or were not of interest, it would be possible for the system to build a profile of "interesting" occurrences, such that it could screen progres-

sively more occurrences of the key word on its own.

These questions provide an initial framework for the evaluation and comparison of the various natural language systems currently under development.

4. Information Capture via Natural Language Understanding

The second major area of application is information capture: the ability to use natural language processing techniques to convert free-form textual information into a database or knowledge base. Such an application is sketched in section 4.3 of the Texas A&M Final Technical Report where NLP was used to process formal information model statements in the IDEF framework [3].

There is a long history of research in this area, although information capture is a substantially more difficult task than query processing, due to the greater syntactic variety, including (for messages) telegraphic style, the necessity of handling referring expressions, and the need to handle discourse phenomena, such as discourse coherence, time, reference resolution, and recovering contextually implicit information.

Some of the early efforts in the area of text processing were done by the Linguistic String Project at New York University, which investigated processing of various types of medical reports [29, 30, 8]. Successors to this work include a joint Unisys-NYU effort, focused on the processing of Navy maintenance reports (CASREPs)[7, 25]. Other efforts in this area include Schank's research [31], the NOMAD system developed by Granger at UC Irvine [6], Lebowitz' *Researcher*

system[17], and Logicon's MATRES system[23].

These systems all share some features, at least in terms of input/output specifications. Input is free text; output is a representation of the information in terms of (quantified) predicates with arguments, represented in terms of case-frames, e.g., a verb followed by its (thematic role) arguments, as in *repair(agent(engineer),patient(compressor))*. In systems that handle time, there is generally the concept of an event or state, with an associated predicate-argument structure and a time (span)[26].

The possible applications of such processed text are numerous. Just as natural language queries can be mapped into DB retrieval requests, predicate argument structures can be converted in DB update requests (although handling of quantification and temporal modifiers raises some issues here). Other approaches include summarization[19] and simulation, and generation of index terms for information retrieval.

5. Mixed Initiative Dialogue

The third (and most sophisticated) type of human-machine interaction is the "mixed initiative dialogue", where machine and human collaborate in a conversational mode to achieve a particular goal [10].

This kind of system can be seen as an outgrowth of other scenarios discussed earlier. For example, truly co-operative response, even in something like

a DB query setting, would consist of a user interacting with (a machine version of) an expert DB administrator, who would aid the user (and possibly instruct the user) in how to obtain the particular information desired.

To achieve this kind of interaction, a range of capabilities are needed. For example, text generation will clearly be involved as soon as there is any kind of "conversation". It will be particularly important in providing explanations or instructions to the user.

A key element of task-oriented dialogue is the notion of progression from some initial state to a final state where a given task has been completed. The system must maintain a model of the changes over time, as the task moves from initial state to completion. Somewhat independent of this, the user will be learning about the task, especially if the system provides explanation and instruction -- thus the machine will need to maintain a model not only of the state of task completion, but of information already presented to the user -- that is, a dynamic model of the state of the user.

The concept of user modeling is even more important if the primary purpose of the system is to aid or instruct a range of users. In this case, the system must infer from the user's questions (or perhaps from some preliminary information furnished explicitly by the user) what the user's state of expertise is. This will enable the system to furnish the most relevant amount and depth of information to the user.

Intermediate between a fully automated system that learns as it interacts with the user, and a "dumb" system that cannot adapt to the user is the notion of *machine-aided* task execution. This has been the dominant paradigm, for example, in machine-aided translation systems, but is also applicable to other areas, such as financial transactions, intelligence stations, automatic zip-code assignment, etc. In a machine-aided setting, the machine performs what tasks it can without human help, but recognizes a class of inputs as being outside the set of information supplied. In such a system the NL system would process the inputs and notify the operator of those transactions that it cannot process. The system could supply the information required to identify the transaction and characterize the parameters which it used to recognize it as information it could not process. The operator would then interact to complete the processing (or at least eliminate the problem) and the "new" information would be added to the knowledge base (if appropriate), further extending the operations performed by the system. The interface might use speech synthesis to alert the operator and explain the status, but could then resort to more classic interfaces to accept additional inputs.

In addition to the notion of "mixed initiative dialogue", it is important to mention mixed media interaction, where the system (and the user) select the medium most appropriate to the type of information. For example, the user may want to point to an area on a map (using the mouse), in order bring up a tabular display of data about the area on the map, and then ask a question, in

natural language, about some of the data in the display. Similarly, the system might "answer" a question about the location of an object on a map by *highlighting* the location of the object on the map. This notion of "seamless multi-media" interfaces is an active area of research at several institutions (CMU, ISI).

6. Spin-offs from Natural Language Research

Aside from the types of human-machine interaction explored in the earlier sections, there are a number of areas that have great relevance outside strict natural language applications. These areas deserve special mention, because the research here may have broad applicability even when no natural language interface is involved.

6.1. Temporal Processing

The first of these is the processing of temporal information. Time is obviously a highly visible and important part of communication through language. It is communicated through a variety of linguistic devices, including verb tense (*The bus left* vs. *The bus leaves*), aspect (*The bus has left* vs. *The bus left*), adverbial expressions (*The bus left on Sunday*), verb semantics (*The bus' departure preceded the loading of the cargo*), subordinate clauses (*The bus left after they arrived*) and other related devices, such as causal statements (*The bus stopped because it was out of gas*).

By exploring the ways in which time is expressed in natural language, linguists have uncovered a rich variety of temporal relations [2, 27]. The ability to represent these relations and to update a knowledge base as it changes over time are clearly issues with relevance not just to natural language processing, but to any system that tries to represent a dynamic situation. Change over time also has extremely important implications for the database world, although conventional DB systems do not allow any sophisticated reasoning about temporal relations between events. This will clearly be a major research area for knowledge representation and databases in the new few years.

6.2. User Modeling

The area of user modeling, although mentioned in connection with mixed initiative dialogue systems, clearly is not limited to natural language interaction. The notion of user modeling is essential to the successful design of human-machine interfaces.

There are two dimensions to user-modeling: the modeling of a range of users of a system, in order to tailor responses to the appropriate type of user[16]; and the ability of a given user to learn over time, so that information once given, need not be repeated over and over again: explanations may become more succinct, for example.

The first kind of user modeling, namely detecting the level of a user, is somewhat analogous to fault diagnosis -- the system needs to "diagnose" the

level of the user, based on limited (and sometimes inconsistent) information.

The second aspect of user modeling will need to model the change in the user's state over the course of the interaction. In particular, such a user model needs to maintain its context, where the context consists of previously exchanged information. This will be changing continuously over the course of the dialogue. Failure to incorporate at least limited user modeling will lead to systems that seem very "dumb" and repetitious, as well as inflexible or unfriendly.

6.3. Tools for Global Semantics

The ideal user interface to a distributed, heterogeneous database system would provide the user with a uniform view of the information contained in the system: a global DB semantics. The construction of such a global semantics is a complex task; it involves reconciling different views of the data, different implicit semantics, different units of measure, etc.

Since natural language interfaces must also model a kind of global semantics, which relates language constructs to general semantic classes and relationships in the domain, the tools developed for modeling semantics in natural language may well provide a useful "global semantics" for heterogeneous databases. One useful strategy may be to define two levels of semantics -- a general semantics, used for supporting the natural language interface, and a second "global DB semantics" which is defined by a mapping from the general concepts

to those concepts and relationships supported by specific the distributed databases. The relationship between the semantics needed for natural language and the kind of semantics needed for a global DB semantics is an interesting open research issue.

6.4. Knowledge Acquisition

At the heart of both expert systems applications and natural language applications is the issue of knowledge acquisition: how to get the domain-specific knowledge into the system. It is clear that advances in this area will also affect the DB area, since the same techniques used in expert systems can be used to structure databases and to input information into databases. Section 4 discussed information capture using natural language techniques, but based on the assumption that one already had a natural language system running in a given domain. Here the issue is: how to get that system running.

For natural language systems, it is necessary to acquire both linguistic and general domain knowledge. There is extensive research now on building a range of tools to aid in this process. The first level of tool is a knowledge representation framework that permits the user to model various "second order" relationships in a domain, e.g., part-whole, and type relationships. A second level tool would prompt the user to generalize this information, and would check for missing and inconsistent information. More advanced tools might perform generalizations and analogies on their own, given novel problems and sets of data, to provide limited "machine learning". The success of expert systems technology

long-term (and also natural language interfaces) will depend heavily on the availability of such tools to ease the knowledge acquisition process.

7. Conclusion

The preceding sections have discussed a variety of uses for natural language interfaces, as well as some tools that continued research in this area will make available. This section will provide a brief discussion of the timetable for such results.

7.1. Information Access

There are now commercial systems providing limited natural language access to databases. The language coverage is not very broad and the systems are limited in their ability to flag and explain their errors. However, this technology is currently available and will continue to improve. It is clear that natural language interfaces have not created a sudden demand for more such systems. This is because typing natural language into a system, though perhaps more friendly for the naive user than a DB retrieval request, is still cumbersome. For natural language interfaces to become truly successful, it will be necessary to provide speech recognition, so that a user can *speak*, rather than type, to get information out. Speech recognition is still a very fragile technology, though enormous gains have been made in signal processing hardware and software. It seems likely that as researchers link the signal processing back-end

with the linguistic/semantic frameworks developed for text processing that we should begin to see reasonable continuous speech recognition (in limited domains) in the next 4-8 years.

7.2. Information Capture

There are only very limited "custom systems" available today for information capture from natural language (for example, the bank telex system ATRANS, developed by Cognitive Systems[18]. However, this technology is maturing and over the next few years, there should be a number of custom systems developed to capture information from various types of messages. We are also seeing the first natural language "toolkit", namely Carnegie Group's LanguageCraft. Although LanguageCraft leaves much to be desired, it is an interesting first step in providing users with a limited tool set from which to build natural language interfaces (this is mostly aimed at building query interfaces, but has been used to build limited text processing interfaces as well -- as mentioned in the Final Technical Report on this project, section 4.3[3]).

For this class of application, there are still two serious problems: coverage of the system, and the issue of acquiring sufficient domain-specific knowledge. In general, most current systems have a fairly limited coverage of the language; those systems which have a broader coverage have problems with robustness and maintainability, particularly if the system is to be maintained by non-linguists. The portability issue is being addressed in a number of ongoing research efforts, including work by Hirschman[13], Ballard[4], Moser[24], among

others. However, tools in this area are still quite primitive and require substantial expertise on the part of the user. It will still be three to five years before we see good sets of tools to support portability of large scale systems.

It seems likely that we will see a steady growth in custom applications and an increasing number of "toolkits" over the next three to five years. In this area, there is less of a need to couple the system to speech recognition, since there are huge volumes of written material routinely transmitted; however, the ability to accurately capture spoken language would revolutionize various applications. For example, the introduction of a "talking typewriter", which transcribes spoken input, would have an significant impact on the office automation area. The commercial market here is enormous, and we are already seeing the first attempts at introducing such products.

7.3. Mixed Initiative Dialogue

The mixed initiative dialogue places the greatest demand on the system, to not only understand, but to generate reasonable responses. In the past, generation has been handled largely by canned or template-driven responses. However, there is substantial research going on in text generation, and there are now research systems capable of doing generation for specific tasks [21, 20]. However, these systems are still research systems, and it would require substantial work to turn them into a commercially marketable "toolkit". The obvious next step is to couple generation systems with speech synthesis systems, to provide spoken output; these advances will being taking place in the next year or two.

The area of mixed initiative dialogue will prove to be an extremely useful one as expert systems applications become more widespread. Such systems will also play a major role in computer-aided instruction. Again, the addition of speech recognition and generation will probably make an enormous difference in user acceptance and utility of such systems.

In general, the next five years to seven years should prove to be a pivotal period in the deployment of natural language interfaces. As speech technology develops to the point where spoken language can become the medium of interaction, the demand for such interfaces will probably increase dramatically. It is clear that the technology is maturing to the point where it will be able to provide order of magnitude improvements in the human/machine interface, and any system of the future will almost certainly include the ability to interact with the system via natural language, both written and spoken.

8. REFERENCES

[1] A. Meyers , VOX -- An Extensible Natural Language Processor. In *Proceedings of IJCAI-85*, Los Angeles, CA. , 1985, pp. 821-825..

[2] Maintaining Knowledge about Temporal Intervals. *Comm. of the ACM* **26**, 11 , 1983, pp. 832-843.

[3] Knowledge Based Systems Laboratory, Texas A&M, Knowledge Based Integrated Information Systems Development Methodologies Plan, Final Technical Report, College Station, TX, May, 1987.

[4] B. Ballard, TELI. In *Proc. of the 24th Annual Conference of the Assoc. of Computational Linguistics*, 1986.

[5] T.E. Doszkocs and B.A. Rapp, Searching MEDLINE in English: A Prototype User Interface with Natural Language Query, Ranked Output, and Relevance Feedback. In *Proceedings of the ASIS Annual Meeting*, Knowledge Industry Publications, 1979, pp. 131-139.

[6] R.H. Granger, C.J. Staros, G.B. Taylor, and R. Yoshii, Scruffy Text Understanding: Design and Implementation of the NOMAD System. In *Proceeedings of the Conference on Applied Natural Language Processing*, 1983, pp. 104-106.

[7] R. Grishman and L. Hirschman , PROTEUS and PUNDIT: Research in Text Understanding. *Computational Linguistics* **12**(2), , pp. 141-145.

[8] R. Grishman and L. Hirschman, Question-Answering from Natural Language Medical Data Bases. *Artificial Intelligence* **11**, 1978, pp. 25-43.

[9] Barbara Grosz, TEAM: A Transportable Natural-Language Interface System. In *Proceedings of the Conference on Applied Natural Language Processing*, Santa Monica, CA, February, 1983, pp. 39-45.

[10] B. Grosz, Focusing and Description in Natural Language Dialogues. In *Elements of Discourse Understanding*, A. Joshi, B. Webber, and I. Sag (ed.), Cambridge University Press, Cambridge, 1981, pp. 84-105.

[11] A. Gupta and S. Madnick, Executive Summary, Knowledge-Based Integrated Information Systems Engineering, Sloan School, MIT, Cambridge, MA, May 1987.

[12] L. Hirschman, Guy Story, Elaine Marsh, Margaret Lyman, and Naomi Sager, An Experiment in Automated Health Care Evaluation from Narrative Medical Records. *Computers and Biomedical Research* **14**, 1981, pp. 447-463.

[13] L. Hirschman, Discovering Sublanguage Structures. In *Sublanguage: Description and Processing*, R. Kittredge and R. Grishman (ed.), Lawrence Erlbaum Assoc., Hillsdale, NJ, 1986.

[14] K. Jensen, G.E. Heidorn, L.A. Miller, and Y. Ravin, Parse Fitting and Prose Fixing: Getting a Hold on Ill-Formedness. *Computational Linguistics* **9**(3-4), 1983, pp. 147-160.

[15] J.K. Kalita, M.L. Jones, and G.I. McCalla, Summarizing Natural Language Database Responses. *Computational Linguistics* **12**(2), 1986, pp. 107-124.

[16] R. Kass and T. Finin, Rules for the Implicit Acquisition of Knowledge about the User. In *Proc. of AAAI-87*, Seattle, 1987.

[17] M. Lebowitz, Researcher: An Experimental Intelligent Information System. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1985, pp. 858-862.

[18] Steven Lytinen and Anatole Gershman, ATRANS: Automatic Processing of Money Transfer Messages. In *Proc. of AAAI-86*, Philadelphia, PA, 1986, pp. 1083-1088.

[19] E. Marsh, H. Hamburger, and R. Grishman, A Production Rule System for Message Summarization. In *Proc. 1984 National Conf. on Artificial Intelligence*, Oakland University, Rochester, Michigan, 1984.

[20] David D. McDonald and James Pustejovsky, Tags as a Grammatical Formalism for Generation. In *Proc. of the 23rd Annual Meeting of the ACL*, Chicago, 1985, pp. 94-103.

[21] Kathleen R. McKeown, . In *TEXT GENERATION: Using Discourse Strategies and Focus Constraints to Generate Natural Language*, Cambridge University Press, 1985.

[22] L.A. Miller, G.E. Heidorn, and K. Jensen, Text-critiquing with the EPISTLE System: An Author's Aid to Better Syntax. In *Proc. Nat'l Comp. Conf.*, AFIPS Press, Arlington, VA, 1981, pp. 649-655.

[23] C. A. Montgomery, Distinguishing Fact from Opinion and Events from Meta-Events. In *Proc. Conf. on Applied Natural Language Processing*, Santa Monica, CA, February 1983, pp. 55-61.

[24] M. G. Moser, Domain Dependent Semantic Acquisition. In *Proc. of the First Conference on Artificial Intelligence Applications*, Denver, CO, 1984, pp. 13-18.

[25] Martha S. Palmer, Deborah A. Dahl, Rebecca J. [Passonneau] Schiffman, Lynette Hirschman, Marcia Linebarger, and John Dowding, Recovering Implicit Information, Proc. of the 24th Annual Meeting of the Association for Computational Linguistics, Columbia University, New York, August 1986.

[26] Rebecca J. Passonneau, Situations and Intervals, Accepted for presentation at the 25th Annual Meeting of the Assoc. for Computational Linguistics, Stanford, July, 1987.

[27] Rebecca J. Passonneau, A Computational Model of the Semantics of Tense and Aspect. *Journal of Computational Linguistics*, May, 1987.

[28] L. M. Reeker, E. M. Zamora, and P. E. Blower, Specialized Information Extraction: Automatic Chemical Reaction Coding from English Descriptions. In *Proc. Conf. on Applied Natural Language Processing*, Santa Monica, CA, February 1983, pp. 109-116.

[29] Naomi Sager, Natural Language Information Formatting: The Automatic Conversion of Texts to a Structured Data Base. In *Advances in Computers*, M.C. Yovits and M. Rubinoff (ed.), Academic Press , New York, 1978, pp. 89-162.

[30] N. Sager, *Natural Language Information Processing: A Computer Grammar of English and Its Applications*. Addison-Wesley, Reading, Mass., 1981.

[31] R.C. Schank, J.L. Kolodner, and G. DeJong, Conceptual Information Retrieval. In *Information Retrieval Research*, R.N. Oddy et al. (ed.), Butterworth and Co. Ltd., London, 1981, pp. 94-116 .

[32] M.S. Tuttle, D.D. Sherertz, M.S. Blois, and S. Nelson, Expertness" form Structured Text? RECONSIDER: A Diagnostic Prompting Program. In *Proceeedings of the Conference on Applied Natural Language Processing*, 1983, pp. 124-131.

[33] G. Vladutz, Natural Language Text Segmentation Techniques Applied to the Automatic Compilation of Printed Subject Indexes and for Online Database Access. In *Proceeedings of the Conference on Applied Natural Language Processing*, 1983, pp. 136-142.

[34] R. Weischedel et al., , BBN Laboratories: Research and Development in Natural Language Processing in the Strategic Computing Program. *Computational Linguistics* **12**(2), 1986, pp. 132-136.

COMPUTER-AIDED SUPPORT OF DELIBERATION AND JUDGEMENT: A PROCESS-ORIENTED APPROACH TO DSS DESIGN

J.F. Nunamaker

University of Arizona

Lynda M. Applegate

Harvard University

Benn R. Konsynski

Harvard University/University of Arizona

Contributed by J. F. Nunamaker

COMPUTER-AIDED SUPPORT OF DELIBERATION & JUDGEMENT: A PROCESS-ORIENTED APPROACH TO DSS DESIGN

by

J.F. Nunamaker
Department of Management Information Systems
University of Arizona

Lynda M. Applegate
Harvard Business School
Harvard University

Benn R. Konsynski
Harvard Business School (Visiting)
On leave from University of Arizona

"It seems obvious that we cannot solve present day major political and organizational problems simply by grinding through a mathematical model or computer algorithm. What we require besides is the design of better deliberation and judgement. Once we begin to understand the process of deliberation and judgement we may converge on a better objective method: that is a way to express optimal deliberation in a precise warranted form".

Churchman & Eisenberg, 1969

1.0 INTRODUCTION

The above quote, written over fifteen years ago, expresses the need for better support of deliberation and judgement to enable more structured problem-solving and decision-making. Despite active interest in this area, researchers continue to call for research that will improve understanding of the decision process (Phillips, 1984; Stabell and Fuglseth, 1985; Little, 1986). These researchers recommend that the design of future decision support systems (DSS) focus on providing support for the decision-making process instead of support for isolated decision tasks. This capability is critical to enable computer-based decision support for strategic decision-making and planning, decision processes characterized by their unstructured nature

(Ackoff, 1970; Gorry and Scott Morton, 1971). Stabell and Fuglseth (1985) state that a decision-oriented approach to the design of DSS is necessary to fully implement the original design for DSS proposed by Gerrity (1971) and Gorry and Scott Morton (1971).

The requirements for supporting unstructured decision processes have been summarized as: (1) access to a wide range of ad-hoc data both internal and external to the organization; (2) flexible access to both quantitative and qualitative decision models; (3) the ability to capture and store decision process knowledge in a flexible knowledge management system that enables identification and maintenance of linkages among interrelated pieces of decision information; and (4) the capability for both group and individual decision support (Applegate, Konsynski and Nunamaker, 1987).

The purpose of this paper is to describe research on the design, implementation and evaluation of an automated system to support complex, unstructured group decision processes within organizations. Strategic planning, a complex, unstructured decision process, was chosen as the domain of the system. A group decision support system (GDSS), the Plexsys Planning System, has been designed for use by top-level managers and planners in a decision laboratory to conduct actual strategic planning sessions for their organizations. As the managers carry on the planning/strategic decision-making process, the decision laboratory simultaneously serves as a research environment for the study of the nature and dynamics of the deliberation and judgement processes and the influence of technology on those processes.

This research builds upon the efforts of past research in the design of the Plexsys system, especially the (1) development of the Plexsys model and vision for the Plexsys system (Konsynski and Nunamaker, 1982), (2) design of the original semantic inheritance network Plexsys Knowledge Management system developed on the VAX 11/780 (Kotteman, Konsynski, Nunamaker and Stott, 1984), and (3) the use of the original Plexsys Knowledge Management system to model information systems planning (McIntyre, Nunamaker and Konsynski, 1986).

2.0 EVOLUTION OF DSS

Automated support for managerial decision-making has been a focus of research efforts since the early 1970s (Gorry and Scott Morton, 1971; Gerrity, 1971). Decision support systems (DSS), originally termed management decision systems by Scott Morton and man-machine decision systems by Gerrity, were defined as "interactive computer-based systems that help decision-makers use data and models to solve unstructured problems" (Scott Morton, 1971). A framework for the design of DSS was proposed consisting of three major components: a data component, a dialogue component and a modeling component (Sprague, 1980). The modeling component was identified as the primary component that transformed a traditional data processing (transaction-oriented) system to a DSS.

Until recently, however, the emphasis of automated decision support in organizations has been primarily to support specific management tasks for individual decision-makers. These tasks are frequently structured to semi-structured in nature and depend primarily on quantitative models for their solution. In the design of early DSS

the data, dialogue and modeling components of the system were tightly coupled and problem-specific. These highly specialized, self-contained systems lacked flexibility and were poorly integrated into organizational information systems.

Major problems identified in these early designs were (1) the systems were time-consuming and costly to develop and maintain resulting in static systems that were difficult to adapt to rapid changes in an organization's decision support requirements, and (2) the systems were highly user-and problem-specific and provided a single model designed to solve a single problem (Naylor and Schauland, 1976). The support provided by early DSS was limited to the structured, recurring tasks within an organization. The ability to provide support for complex, unstructured decision processes, the original vision for DSS, was severely limited.

Since the late 1970s DSS researchers, aided by rapid advancements in information technology, have sought ways of improving automated support for decision-making in organizations. Two major areas of research can be identified. The first research area involved the development of DSS tools and DSS generators that decrease the cost and time necessary to implement and maintain single model, single problem DSS. These tools and generators include spreadsheets (e.g., LOTUS 123, SUPERCALC), modeling languages (e.g., IFPS, MODEL), special purpose DSS design systems (e.g., GADS) and general system design tools (e.g., graphics packages, 4th generation languages). Coupled with the rapid increase in the use of microcomputers and personal workstations, these

software tools have resulted in the widespread use of DSS within organizations.

Data from surveys of large corporations indicate that the design and use of computer models has become a common tool in U.S. businesses. A survey conducted in 1968 by Gershefski (1969) revealed that only 20% of the 323 firms surveyed were using automated decision models. A 1975 survey found that 73% of 346 firms surveyed were developing or using automated decision models (Naylor and Schauland, 1976). A more recent survey by Kein (1982) reported that 85% of 204 companies were using automated models to support decision-making.

The involvement of top management with successful modeling efforts was a consistent finding. Upper managers participated in the definition and implementation of corporate models in 50% - 90% of the companies using models (Naylor, 1976; Kein, 1982). The most frequent end-users of models in early studies were management support staffs in strategic planning and finance/accounting departments. But the growing use of microcomputers and personal workstations in organizations has begun to change the pattern of corporate model end-users. A survey of over 1000 organizations by Data Decisions (1983) found that 67% of middle managers and 22% of top managers of non-data-processing departments were using personal computers to support decision-making. The trend toward increased use of automated decision-making models within organizations appears to be well-established with growing acceptance of the use of personal computers and computer models at the middle and upper levels of management.

Until recently the most common applications of automated decision-making models in organizations were financial forecasting, pro-forma financial statements, capital budgeting, market planning and other quantitative (primarily financial/accounting) models. These models were designed to be used by individual decision makers and provided isolated data to support specific decisions. This application portfolio provided critical data for decisions within organizations but did little to support the process of making the critical decisions.

Clearly, DSS have become an integral part of organizations. The DSS designed by DSS generators, however, still suffer from a single problem, single model perspective. While the cost and time needed to design DSS has improved, little has been done to expand the scope of DSS to provide support for complex, unstructured decision problems. In addition, the rapid proliferation in the use of models within organizations led to a growing recognition of the need to manage organizational models in a manner similar to the management of organizational data (Dolk & Konsynski, 1984).

A second major area of DSS research involved the development of generalized DSS designs that allowed for (1) support of multiple problems using a variety of data sources and models and (2) the centralized management of organizational models. Generalized DSS (Nunamaker, Swenson and Whinston, 1973; Bonczek, Holsapple and Whinston, 1981) and model management systems (Will, 1975; Elam et.al., 1980; Konsynski and Dolk, 1982; Blanning, 1982) were proposed to meet the need for a multi-problem, multi-model perspective and centralized management of organizational models. Generalized DSS architectures

function to integrate the data, dialogue and modeling components of a DSS and provide a framework for DSS design. Static system components can be designed with code that does not change from one application to another. This forms the basis of a library of re-usable DSS modules.

Model management systems (MMS) are one component of a generalized DSS architecture that function to organize, classify, access and retrieve models from a model base in a manner similar to a database management system. An MMS is intended to function with a database management system and a dialogue management system for the design of generalized, flexible DSS capable of addressing a broad problem domain and the needs of a variety of users.

Despite the long period of time devoted to research on generalized DSS and MMS, a significant impact on organizational decision support and modeling efforts has not been identified. The most likely cause for the failure to utilize this important class of systems in organizational settings is the fact that until recently, organizations did not perceive a need for model management and generalized DSS support. As previously indicated, the rapid increase in the reliance on DSS and corporate models has been a fairly recent phenomena. Organizations are just now becoming aware of the opportunities and potential of DSS for support of organizational decisions and the need for management of this valuable information resource. There is a growing recognition among DSS researchers that the time is right to re-evaluate the focus of DSS research in light of the original vision for DSS and the current unmet decision support needs of organizations.

It has been recommended that the next step in the evolution of the DSS concept be to design systems that provide support for complex, unstructured decision processes at all levels of the organization rather than just support for specific, structured decision tasks (Phillips, 1984). To accomplish this task, Stabell (1979), Phillips (1984) and Little (1986) call for an emphasis on decision research that will guide the design of decision support systems that capture the essence of the decision process. Fuglseth and Stabell (1985) and Dhar (1986) call for a reliance on knowledge engineering and knowledge-based systems approaches to the design of DSS for support of complex, unstructured decision processes. Gray et.al., (1981), Huber (1982, 1984), Kraemer and King (1984), DeSanctis and Gallupe (1985), Bui and Jarke (1986), Applegate, Konsynski and Nunamaker (1986), Gallupe (1986) and Nunamaker, Applegate and Konsynski (1987) call for the design of group decision support systems to support the group nature of most upper level management decisions. Table 1.1 summarizes the evolution of the DSS concept and the directions of future research.

TABLE 1.1

EVOLUTION OF DECISION SUPPORT SYSTEMS WITHIN ORGANIZATIONS

	PAST DSS	PRESENT DSS	FUTURE DSS
FOCUS OF SYSTEM	Specific decision problem or task	General class of decision problems	Organization decision processes
RANGE OF DECISION PROBLEMS	Structured	Structured to semi-structured	Full range of decision problems (structured to unstructured)
LEVEL OF ORGANIZATIONAL SUPPORT	Operational control	Operational & management control	Operational & management control & strategic planning
DEGREE OF ORGANIZATIONAL INTEGRATION	Organizational isolation	Beginning organizational integration	Advanced organizational integration Organization-wide perspective
GOAL OF DECISION SUPPORT	Increased decision-making efficiency	Increased decision-making efficiency	Increased decision-making effectiveness & efficiency
DECISION MODELS SUPPORTED	Quantitative analysis models	Quantitative analysis models	Quantitative & qualitative analysis models
DECISION-MAKER SUPPORTED	Individual	Individual	Individual & group
SYSTEM DESIGN STRATEGY	Isolated, stand-alone systems; Models imbedded into code; Single model, single problem perspective	Isolated, stand-alone systems predominate; Models created on DSS generator shell; Beginning multiple model, multiple problem perspective	Generalized, integrated systems; Models and model fragments separated from decision processing code; Multiple model, multiple problem perspective

Research on the design and implementation of DSS that satisfy the requirements for future automated decision support is just beginning. The Plexsys Planning System has been developed to meet the challenge of designing an automated system that provides support for organizational decision processes by integrating data (internal and external), models (quantitative and qualitative) and process management within a group decision support focus. The strategic planning process has been chosen as the domain of the system to demonstrate the ability of the architecture to support complex, unstructured decision processes frequently found at the upper levels of management within organizations. For the purposes of this paper, one component of the domain, the stakeholder identification and assumption surfacing process has been chosen to demonstrate the technical and operational feasibility of the system. The use of the system for support of the idea generation and analysis process has been described in previous work. (Applegate, 1986).

2.0 CONDUCT OF THE RESEARCH

The design of the Plexsys Planning System was based upon study of the planning process conducted in three organizations. Action research methods (Evered and Sussman, 1978; Argyris, 1978;) were used to evaluate the structure of planning decisions within the organization and the dynamics of the strategic planning process. Action research has been identified as the research methodology of choice for studying the dynamics of the decision process and the design of decision support systems (Keen and Scott Morton, 1978; Gibson, 1975; Benbasat, 1984). Data collected during this phase of the research can be found in

Nunamaker and Konsynski (1979), Nunamaker and Applegate (1985); and Applegate, Mason and Thorpe (1986). These data were used to define the requirements of the Plexsys Planning System. A prototype system was developed and implemented in the MIS Planning and Decision Laboratory at the University of Arizona. The MIS Planning and Decision Laboratory (Figure 1) is a state-of-the-art facility where groups of planners and decision-makers from private and public firms can assemble to conduct actual decision-making and planning sessions. The laboratory includes a U-shaped table with recessed, networked microcomputers, large screen projection, break-out rooms with microcomputers on the local network and central file and process servers. Planning session facilitators, chosen for their knowledge and skill in organization development and information systems technology, are available to assist the group in their planning activities.

The network provides access to a wide variety of qualitative and quantitative strategic planning and decision aids for support of idea generation and analysis, market analysis, competitor analysis, strategic forecasting, information systems planning and other organization planning and decision activities. Automated multi-criteria decision-making and voting tools assist the group in evaluating and selecting strategic options. The use of the automated technology is integrated with interactive group discussion to enable face-to-face discussion of issues. The need for face-to-face discussion in decision-making and planning has been identified as

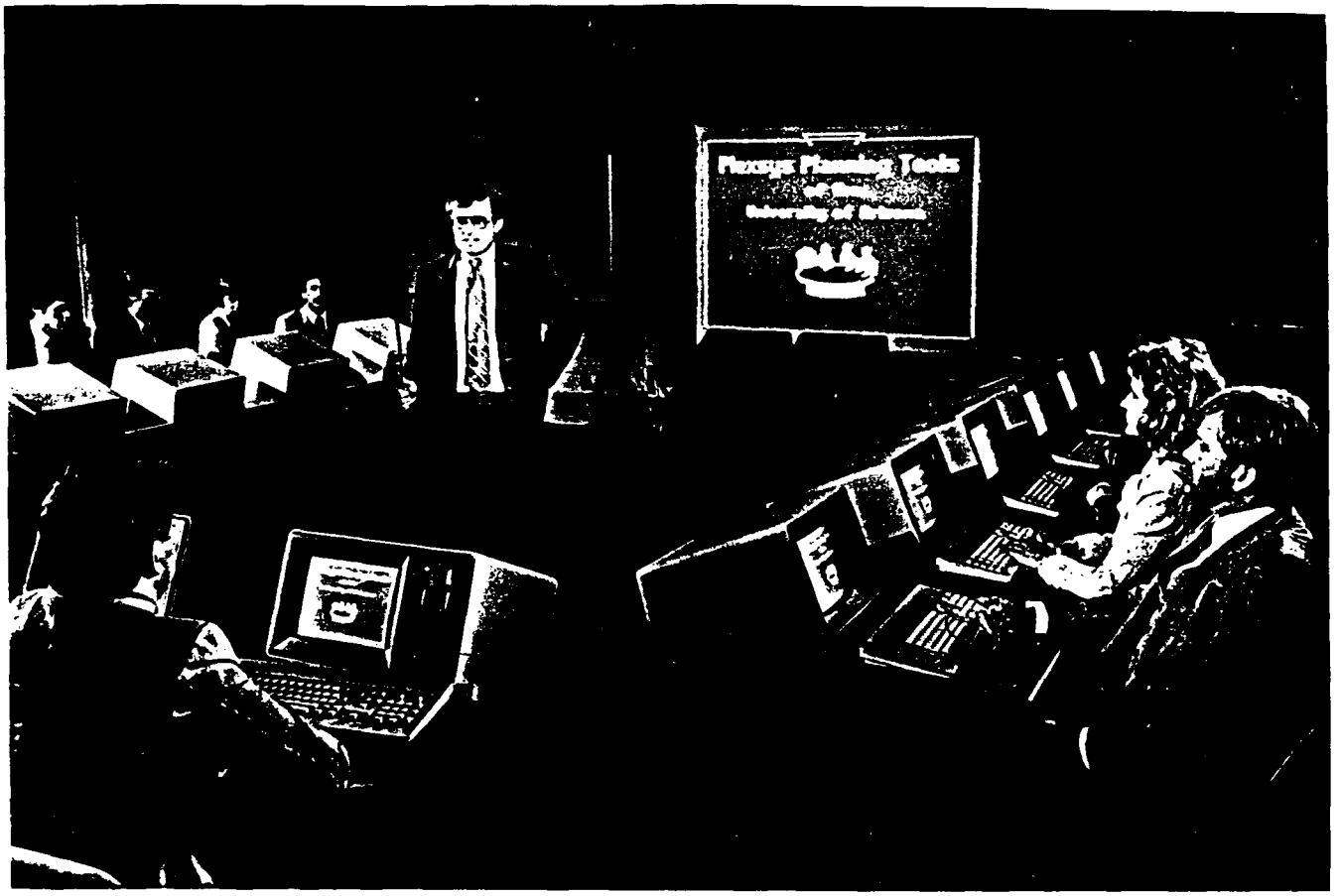


Figure 1

MIS Planning and Decision Laboratory

critical for facilitating group consensus and negotiation during computer communication (Hiltz and Turoff, 1982; Siegel et.al., 1986).

Throughout the planning session, data on the group decision process and the influence of the technology on the decision-making activities are collected and analyzed by software resident on central file and process servers and by trained observers. The data are available for use by decision researchers in studying the unstructured decision-making process at upper levels of management within organizations. In addition, the data are available to the decision-makers to provide valuable insight on the effectiveness and efficiency of their group decision process.

A knowledge management system that enables storage, representation and management of decision information, a process-oriented decision history and the important linkages of decision information collected during a decision process is also provided. The knowledge management system employs a semantic network and frame knowledge representation mechanism implemented using a knowledge representation language (PLEXL) created by the authors and a collection of knowledge management software tools.

The system was originally tested with groups of students conducting mock planning sessions. Revisions to the system were implemented. Technical feasibility, reliability and validity of the current version of the system has been evaluated during use of the system by over 100 planners representing seven planning groups and three different organizations. For the purposes of this paper, a representative planning session from one planning group has been chosen

to provide data to facilitate the description of the Plexsys Planning System. The session chosen is from the 32 Bit Microcomputer Market Analysis case. This case involved a group of planners from a major computer manufacturing company that wished to evaluate the profitability of developing a 32 bit microcomputer product line.

Currently the system is being implemented in a second decision laboratory within the University of Arizona and in a decision laboratory at another university.

3.0 DSS DESIGN FOR STRATEGIC LEVEL DECISION SUPPORT

The Plexsys Plexsys Planning System is designed to meet the functional requirements for automated system support of complex, unstructured decision processes in a group decision support environment. It integrates previous research on individual and group decision support systems, model management systems, executive information systems, and knowledge management systems. It not only provides an example of an integrated system architecture to support these system components but also proposes a new system component -- a Process Management System -- which enables a system focus on the support of decision processes instead of specific decisions or decision tasks.

From the viewpoint of the organizational planners and decision-makers, the Plexsys Planning System is an integrated workbench of planning and decision models that can be used in individual or group mode to (1) retrieve data from internal and external data sources, (2) analyze that data using a wide range of qualitative and quantitative models, (3) generate the critical issues and assumptions on which their

decisions are based, and (4) link these assumptions to the decisions and store them for future reference.

Individuals and groups of planners interact with the system through the use of qualitative and quantitative planning models stored in a centralized model base. Access to the models is controlled using a process management system that, through it's interaction with a centralized knowledge management system, transfers the appropriate models and data for a given strategic planning or decision process to a working storage area for rapid access during a planning session. This paper discusses the Process Management and Knowledge Management systems. An in-depth discussion of the total Plexsys Planning System can be found in Applegate (1986).

3.1 PROCESS MANAGEMENT IN PLEXSYS

The ability to represent, store and manage planning and decision process information in a flexible, dynamic knowledge management system represents an important advance in automated support of organizational decision-making. The Process Management System provides individuals and groups of decision-makers with the tools necessary to (1) access knowledge on planning and decision process categories (e.g., strategic market planning) and the available models and data sources that may be used to support the planning and decision processes in that category, (2) describe instances of a specific planning and decision process and analyze and store them in the knowledge base as an instance of a planning or decision process category (e.g., 32 bit microcomputer market analysis), and (3) describe the critical linkages between intra- and extra-organizational decisions, plans and information.

The Process Management component provides planners and decision-makers with access to specific process information including the inputs and outputs of a process and the models available within the knowledge base that may be appropriate for use in a specific class and/or phase of the decision or planning process. In addition, the process management system software tools assist the user in analyzing and storing planning and decision process information from a specific planning/decision-making session. This creates a process and decision history for the organization. The important linkages of information collected during a planning session and the linkages with other planning/decision-making sessions internal and external to the organization are also identified and stored.

The Process Management system includes the following interactive software tools.

(1) Knowledge Base Query: This tool allows a decision-maker or planner to query the knowledge base for specific classes of decision and planning process stored within it. Once a specific class of planning or decision-making process is identified, the user is questioned whether he/she would like to review a specific instance of the process that is currently stored in the knowledge base or develop a new process instance. If the user states that he/she wishes to review and/or edit a process currently in the knowledge base, the tool queries the knowledge base and displays a list of all instances (cases) related to the chosen process that have been stored in the knowledge base. The user selects the desired case and the knowledge base retrieves the appropriate instance level frame and loads the information into a

working storage area. If the user states that he/she wishes to create a new process instance, the tool retrieves a normative process frame (including inputs, outputs and relevant models and data) and loads the information into a working storage area.

(2) Process Description: This tool allows the user to interactively view information relevant to a process and describe a specific instance of the process. A list of models stored in the knowledge base that may be appropriate for use during individual or group planning and decision-making sessions are displayed along with inputs required for the model and available reports that can be produced by the system. Rules stored within the normative process frame assist the planner or decision-maker in selecting the appropriate models and reports.

Since the Plexsys Planning System has been developed for use during formal group planning sessions in an automated group decision support environment, the planning process is defined as a specific planning session and the Process Description tool has been custom-designed to create a session agenda as an output of its use. (It is important to note that the Process Description tool can be easily changed to enable individual planners to select qualitative and quantitative models to support individual planning processes or to enable groups of planners to support planning meetings or other group planning sessions within organizations.)

The agenda describes the planning and decision models to be used during a session, group interaction periods and the topic of discussion, the start time and duration of each model and/or discussion and the the reports to be distributed. In addition, the start and stop

time for a session, the session facilitator name, the date of the session and the name and title of each participant are also described. Figure 2 presents one screen from the Process Description tool highlighting the creation of a planning session agenda for the first session of the 32 Bit Market Planning case.

The models available for use during the market planning process and the reports that are produced from the models are displayed in the windows marked "TOOLS" and "REPORTS". Miscellaneous activities applicable to all planning sessions held in the MIS Planning and Decision Laboratory are displayed in the window marked "MISC". A planning session facilitator uses the Process Description tool to build an agenda for a planning session by positioning the cursor over the model or report needed and pressing F1. Editing commands are also provided that allow for (1) insertion of a model, report or activity into the middle of an agenda, (2) deletion of a model, report or activity from an agenda, and (3) resetting the time allowed for running a model or conducting an introduction or discussion.

Other commands allow for review of the agenda to insure that normative rules available in the knowledge management system concerning the creation of a planning session agenda for the given category of planning process and/or the selection of models and reports to support the planning process, are followed. At the present time, the activation of a rule only results in the notification of the planning session facilitator or planners that a rule has been violated. The violation and steps taken, if any, to correct it, along with the

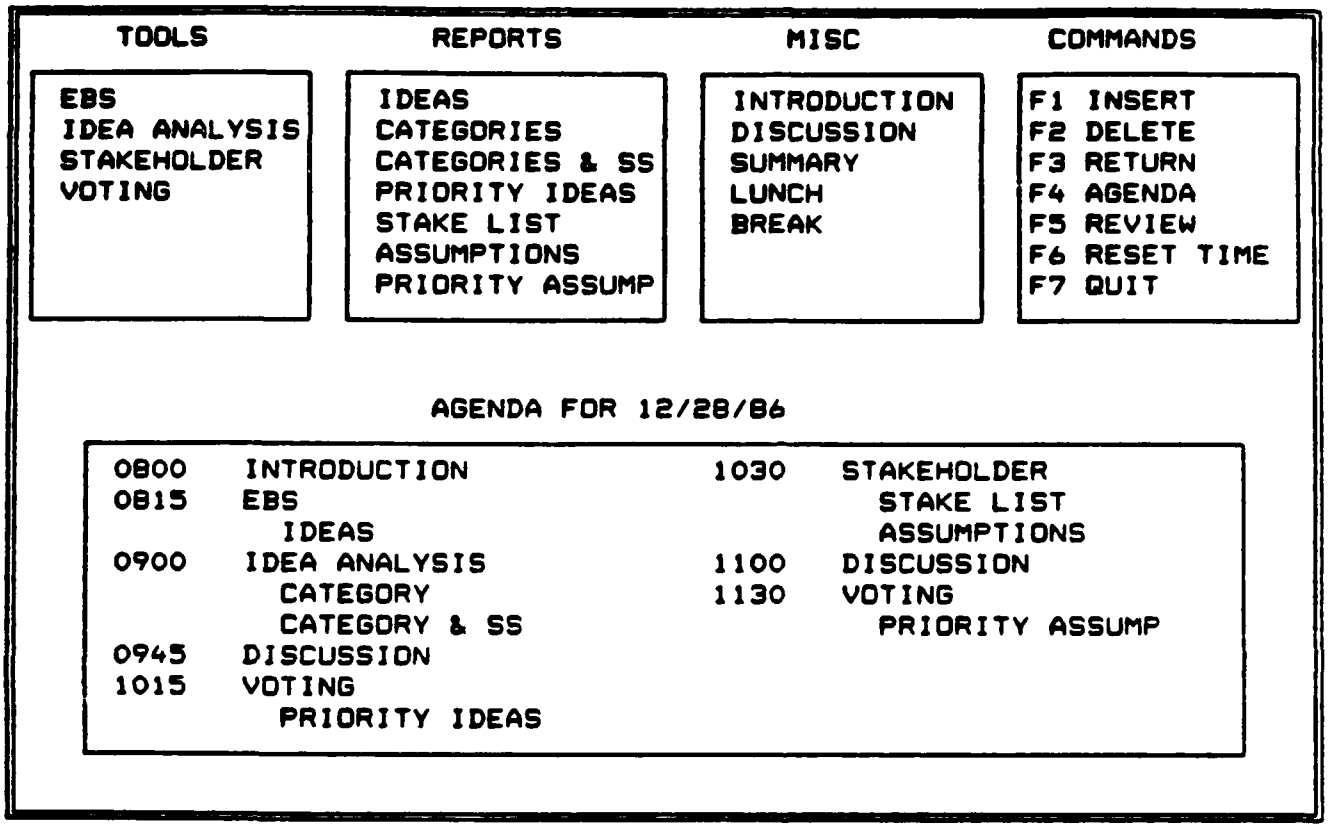


Figure 2

Plexsys Planning System
Process Description Tool

sequence of events followed by the facilitator in establishing the planning agenda are then stored in the knowledge base to allow for automatic recording of the decision process of the facilitator as he/she selects planning models and reports and structures the design of the planning session.

3. Process Manager: Once a process has been defined, the process manager tool accesses the model base and internal and external data bases and retrieves the appropriate models and data. These are stored in a working storage area of the system for rapid access by the decision-makers or planners. The Process Manager then configures a customized program that runs the appropriate models in the defined sequence, accessing necessary information as required. This is controlled by the instance frame of the process that was created using the Process Description tool. As the models are run and information is obtained, the Process Manager analyzes the information based on normative model frames that have been accessed from the knowledge base and stored in the working storage area of the system. Frame transformation rules are used to analyze the instance frames corresponding to a planning process category or specific planning model developed during the session to resolve any discrepancies and, finally, to store the new frame in the knowledge base.

As can be seen in the Process Description tool screen displayed in Figure 2, the agenda for the first session of the 32 Bit Microcomputer Market Analysis case indicates that the planning session began with the total group of 12 planners using the Electronic Brainstorming (EBS)

model to generate ideas on critical issues that must be considered in deciding to enter the 32 bit microcomputer market.

Once the issues had been identified by the group, the report entitled "Ideas" was prepared by the system and a hard copy version of the report was distributed to each planner. In addition, a file containing all of the ideas generated by the planners was shipped to each individual planner's workstation. The individual planners then reviewed the list of ideas and used the Idea Analysis tool to develop categories of ideas and to link the specific ideas with the appropriate categories.

Two reports, the Idea Category report and the Category and Supporting Statement report, were generated. These reports were then used during a face-to-face discussion of the ideas and categories of ideas. During the face-to-face discussion the planners could view on a large screen projector the categories of ideas and supporting statements developed by individual planners. (The projector is connected to a network control switch that permits display of information from each computer on the network.) By the end of the discussion a single list of issue categories and appropriate supporting statement links was developed. This file, representing the consensus of the group, was then shipped to each planner to vote on the importance of each issue category. The Priority Issues report was generated by the system and distributed to the planners. These issues were stored in the model base as an instance frame for the 32 Bit Microcomputer Market Analysis case. They were also linked (via the

semantic inheritance network) to the individual model frames that had been used to generate the specific information.

The planners then split into small groups of 4 planners each which met in the breakout rooms to identify important stakeholders to the 32 bit microcomputer market entry decision and the assumptions that each stakeholder had regarding the decision. The Stakeholder Identification and Assumption Analysis model was used to support this process. Each breakout room was equipped with a microcomputer that was linked to the central Plexsys Planning System process and file servers and to the individual workstations located in the central planning and decision room. With the support of a small group facilitator, each group used the Stakeholder Identification and Assumption Analysis model to identify stakeholders and their assumptions and to rate those assumptions along two axes: (1) the importance of the assumption to the stakeholder and (2) the importance of the assumption to the planner. Figure 3 displays a series of screens that provide a detailed look at the Stakeholder Identification and Assumption Analysis model as an example of one planning model that was accessed during the 32 bit Microcomputer Market Analysis case.

The planners began [screen (a)] by identifying stakeholders to the planning process. These are individuals that the planners believed would have a stake in the decision of the company to enter (or not enter) the 32 bit microcomputer market. As can be seen, the list includes stakeholders internal to the organization (e.g., PC Division, Corporate) and external to the organization (e.g., IBM, Intel). (It is

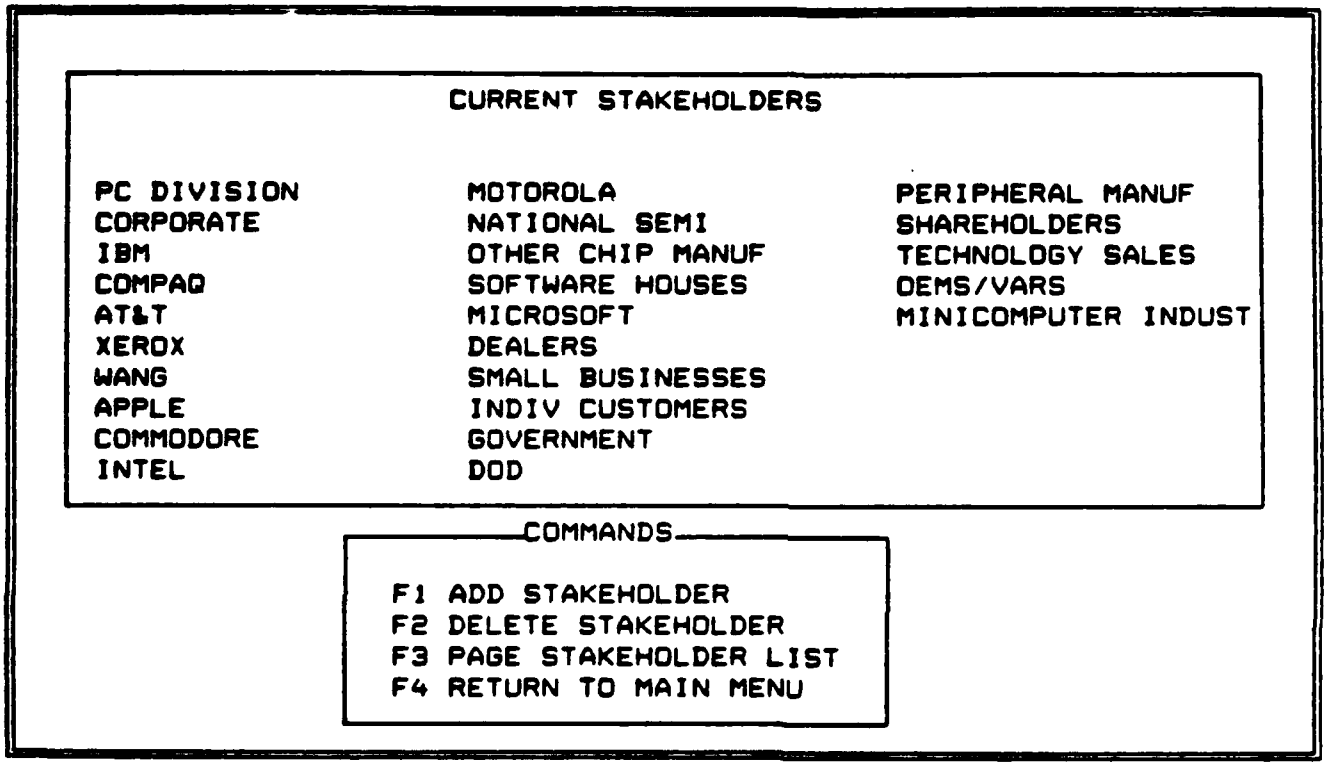


Figure 3
Stakeholder Identification and Assumption Analysis
Screen (a)

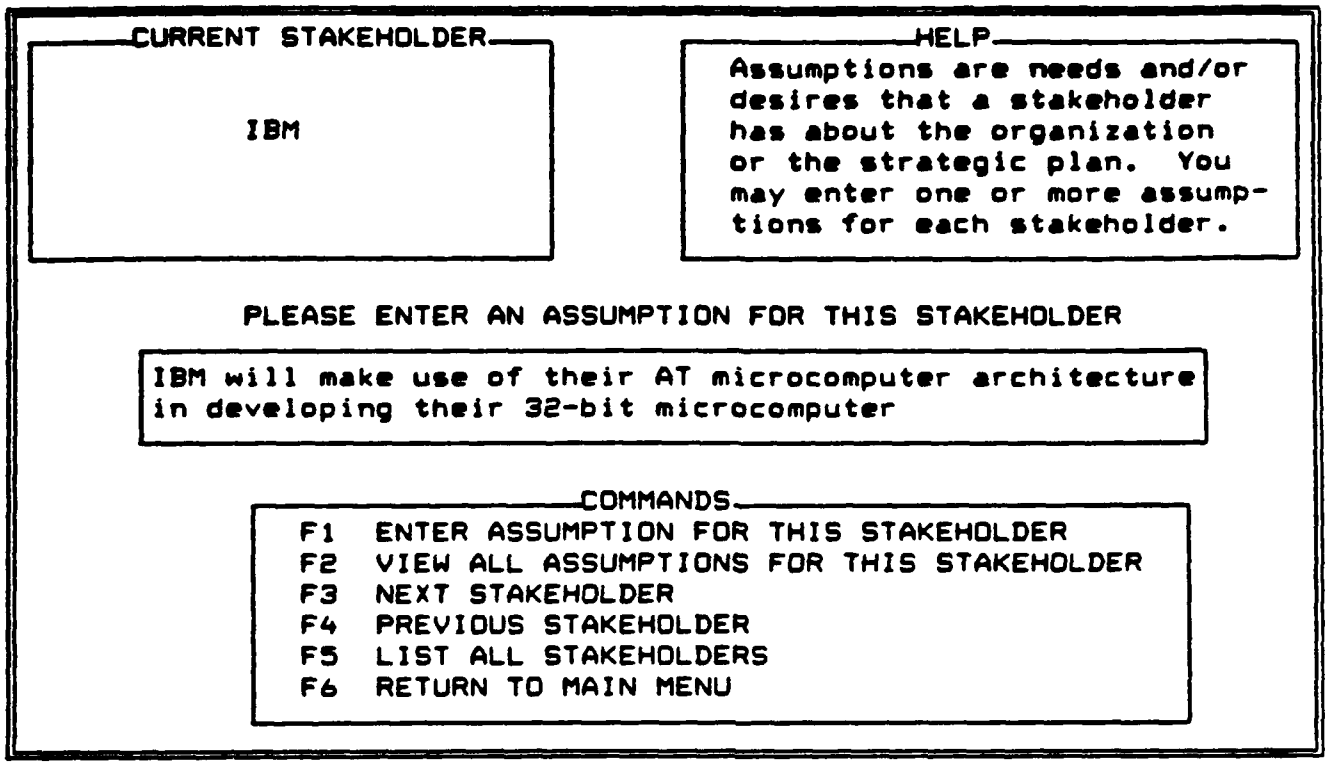


Figure 3
Stakeholder Identification and Assumption Analysis Screen (b)

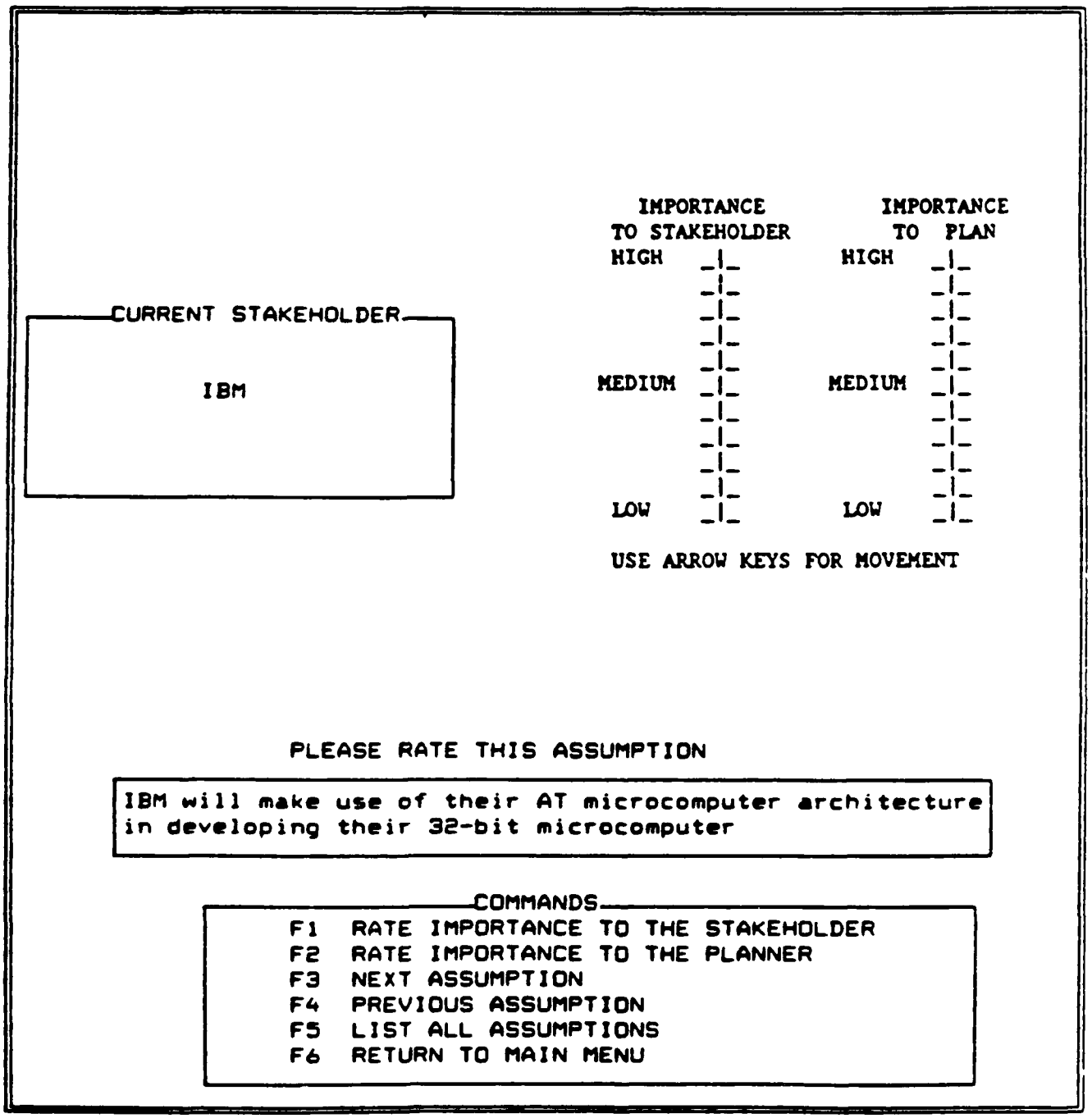


Figure 3
Stakeholder Identification and Assumption Analysis Screen (c)

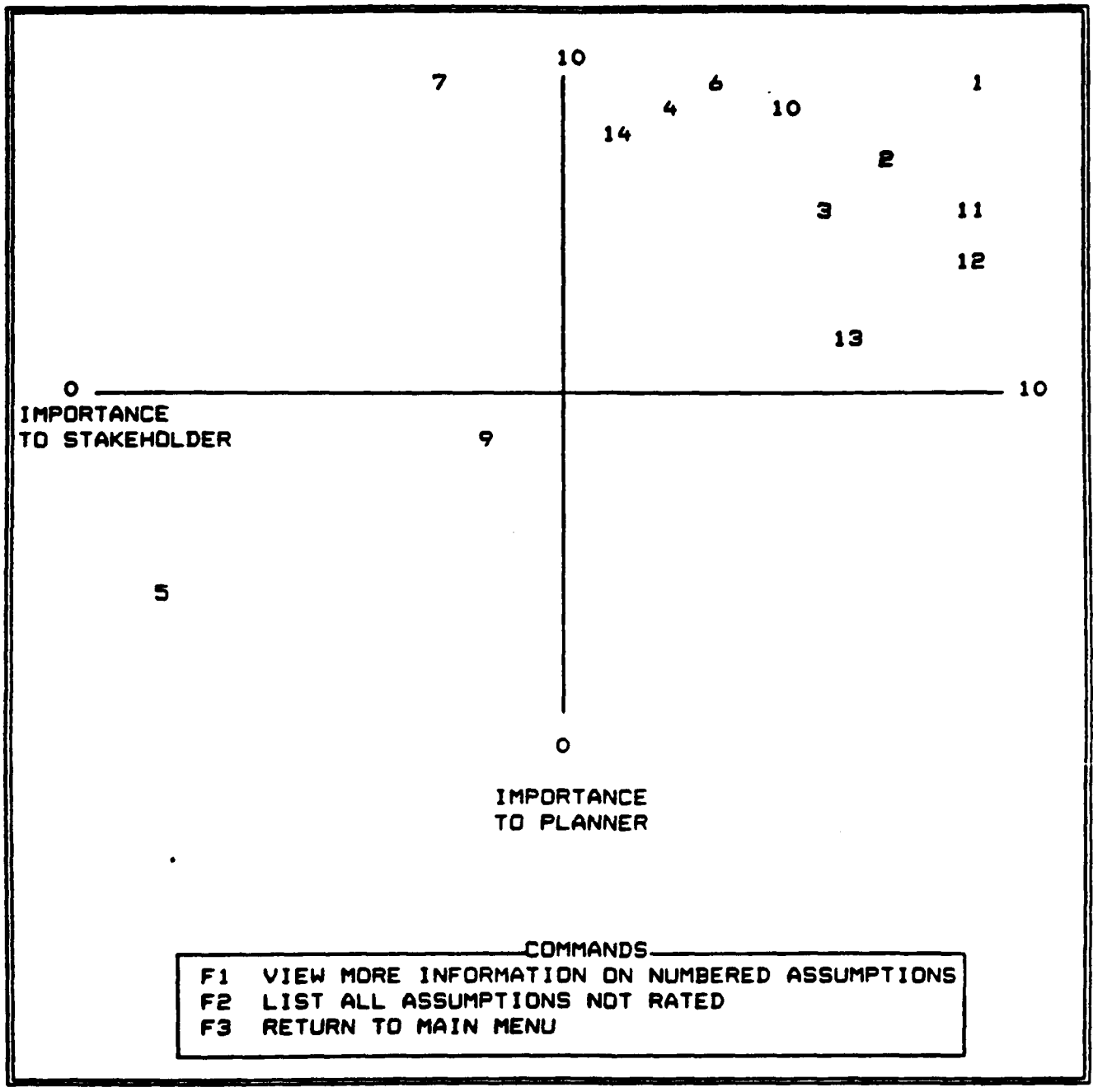


Figure 3
Stakeholder Identification and Assumption Analysis Screen (d)

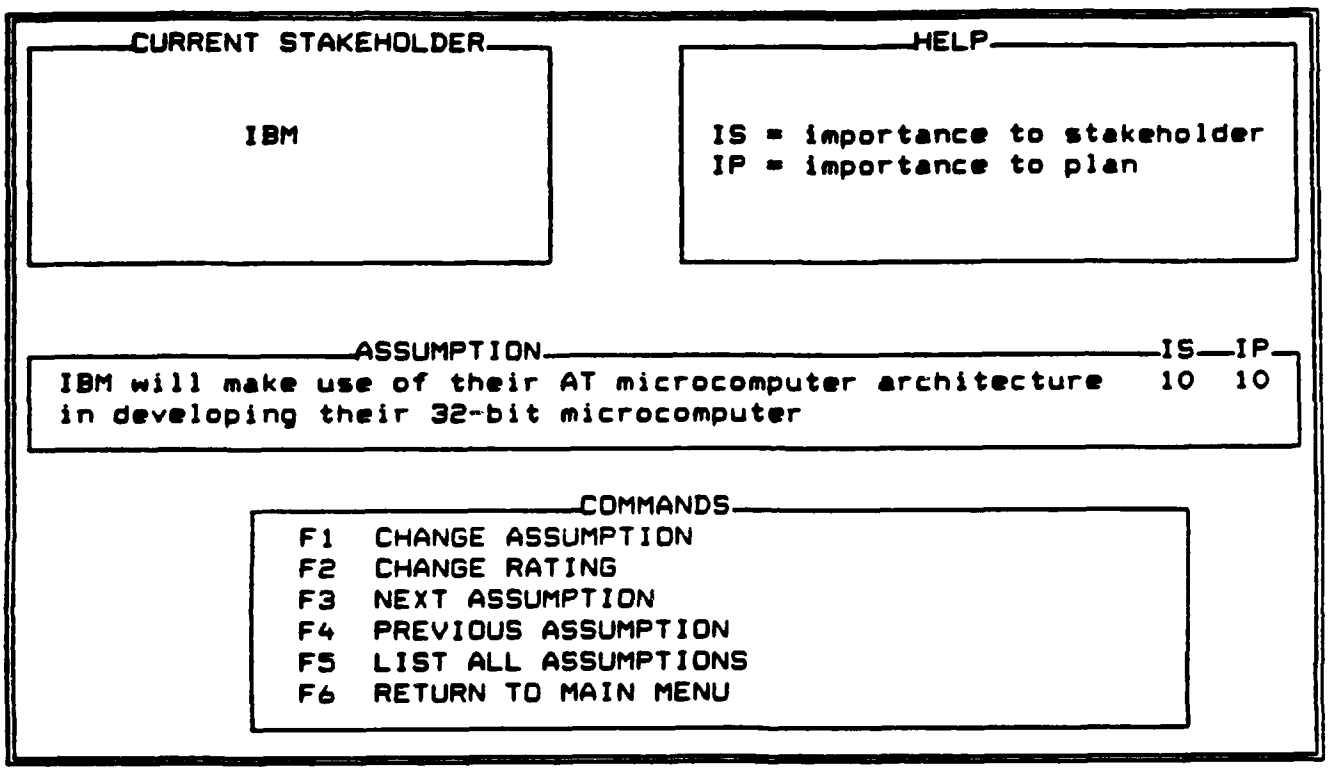


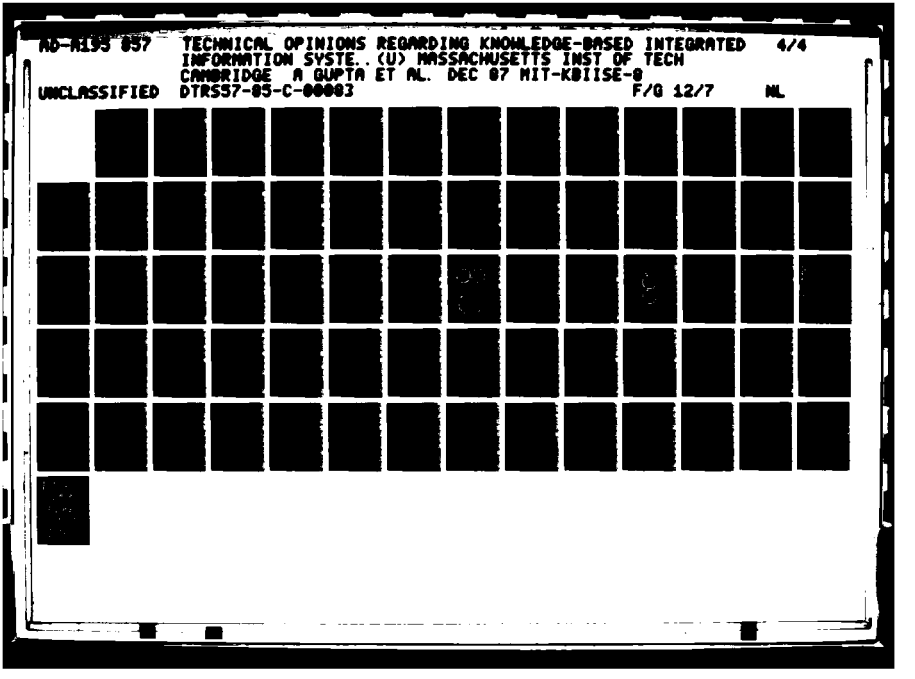
Figure 3
Stakeholder Identification and Assumption Analysis Screen (e)

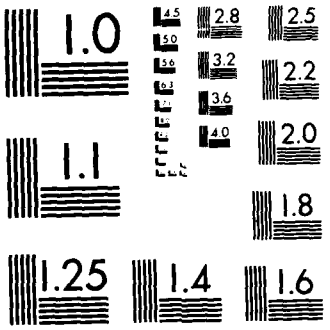
important to note that this list has been recreated from the original list to disguise the company that conducted the planning session.)

Once the stakeholder list was developed, assumptions were entered for each stakeholder. Screen (b) shows the selection of IBM as the stakeholder for whom assumptions would be entered and screen (c) shows the addition of the assumption that "IBM will make use of their AT microcomputer architecture in developing their 32-bit microcomputer". The capability for assigning priority ratings at the time of assumption surfacing is provided but it has been found during use of the system with groups of planners that it is more effective for the planners to rate the assumptions after all assumptions have surfaced.

Once the assumption surfacing and rating process is completed, a graph of the assumptions is provided. [See screen (d).] Each number on the graph signifies that at least one, and maybe more than one, assumption has been rated for planner and stakeholder importance relative to its position on the horizontal and vertical axes. Planners can view assumptions with a given rating by selecting a specific number on the graph. For example, if a planner chose to view assumptions corresponding to number "1" on the graph, they would see all assumptions, one assumption at a time, that had a rating of 10 for importance to the plan and importance to the stakeholder [See screen (e).]

Three reports were generated: (1) a list of stakeholders, (2) the assumptions surfaced for each stakeholder, and (3) the priority assumptions. These reports were presented to all of the planners in hard copy form. In addition, the files for each planning group were


AD-A195 857
TECHNICAL OPINIONS REGARDING KNOWLEDGE-BASED INTEGRATED INFORMATION SYSTE.. (U) MASSACHUSETTS INST OF TECH CAMBRIDGE A GUPTA ET AL. DEC 87 MIT-KBIISE-8
4/4
UNCLASSIFIED
DTRS57-85-C-00003
F/G 12/7
NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

shipped to the server. As a result, the stakeholder and assumption lists and priority analyses developed by each group during the small group discussion period could be projected on the large screen projector for use during the face-to-face large group discussion that followed the small group sessions.

At the completion of the discussion by the total group, the agreed-upon list of stakeholders, assumptions and priority ratings were stored in the knowledge base as an instance of the 32 Bit Microcomputer Market Analysis case and were linked through the inheritance network to the Stakeholder Identification and Assumption Analysis model as an instance frame.

4. Link Manager: These tools were used in a later planning session to link the critical issues (surfaced during the Electronic Brainstorming phase of the planning session) with the lists of stakeholders and priority assumptions (surfaced during the Stakeholder Identification and Assumption Analysis phase of the planning session). These tools are discussed in detail in Applegate (1986).

In summary, the Process Management system functions as an integrated model management and data management system in which process description frames stored within the knowledge base control the retrieval of the appropriate models and data from a centralized model base and internal and external databases. The next section discusses the representation, storage and management of the planning process information in the Plexsys Planning System.

3.2 KNOWLEDGE MANAGEMENT IN PLEXSYS

A fundamental problem that had to be overcome during implementation of the Plexsys Planning System was the determination of a mechanism for representing and storing planning and decision process knowledge within the system. Traditional database management systems, designed to store passive data elements and the relationships among them, would not be appropriate for representing the active elements that comprise process information. A knowledge representation approach has been recommended for representation of modeling information (another active process) within a computer system (Elam et.al., 1980; Konsynski and Dolk, 1982; Applegate, Klein, Konsynski and Nunamaker, 1985; Dhar, 1986) and has been chosen for representation of both process and modeling knowledge in the Plexsys Planning System.

Brachman and Levesque (1985) define knowledge representation as a mechanism for describing the state of the world in such a way that a computer system can draw inferences about a system domain. The current trend is for knowledge to be represented as "explicitly as possible in a formal language". A key component of a knowledge-based system is that the architecture of the system includes explicit knowledge bases that involve direct symbolic encoding of domain-specific knowledge. This is accomplished through the use of a knowledge representation language. The knowledge representation language, specifying explicit system knowledge, only represents one component of a knowledge-based system. An inference mechanism, providing implicit knowledge, and a knowledge domain, from which the explicit and implicit knowledge is drawn, are other important components of a knowledge-based system.

The Plexsys Planning System is a knowledge-based system that addresses all three of these components. Planning process and decision process knowledge are represented and stored within the system using a hybrid knowledge representation technique that is a combination of a semantic inheritance network (Quillian, 1968) and frame representation (Minsky, 1975). The combination of multiple knowledge representation mechanisms is designed to improve the efficiency of the system while providing the necessary representation power. The fundamental tradeoffs in the use of knowledge representation schemes that prompted the choice of the knowledge representation mechanism for the Plexsys Planning System have been discussed in the literature on knowledge representation theory (Levesque and Brachman, 1985) and also in the literature on representation of modeling knowledge in computer systems (Applegate, Klein, Konsynski and Nunamaker, 1985).

Planning and decision process knowledge is stored as a semantic inheritance network that includes attribute links and inheritance links. The semantic inheritance network functions to define planning and decision knowledge as terms and expressions. Terms represent the basic planning and decision objects and the attributes of the objects. Expressions link the planning objects with their attributes and link high-level (abstract) objects with more specific (detailed) objects. A frame representation is then used to provide multiple views of the planning knowledge within the Plexsys Planning System. These frames provide a mechanism for "chunking" planning and decision knowledge to organize and structure the complex, ill-structured planning and decision process domain. Figure 4 presents an abbreviated example of a

Plexsys Planning System semantic inheritance network and frame knowledge representation.

The above diagram illustrates how object terms (e.g., PROCESS, MODEL and STAKEHOLDER IDENTIFICATION & ASSUMPTION ANALYSIS) can be combined with relationship terms (e.g., isa, has, accepts and creates) and attribute terms (e.g., INPUT, OUTPUT, MODEL and SUB-MODEL) to form expressions. Two types of expressions are illustrated. Attribute expressions (e.g., PROCESS accepts INPUT) are used to define an object term. Frames are defined for object terms through the definition of attribute expressions for an object term. Production rules can also be defined within the frame to provide system knowledge about the frame.

Inheritance expressions (e.g., STAKEHOLDER IDENTIFICATION & ASSUMPTION ANALYSIS isa MODEL) are used to create inheritance links between terms and term frames. The inheritance expressions allow frames defined on one level of the system to be inherited by a second term.

A knowledge representation language has been developed by the authors for explicitly specifying knowledge within the system. This language is the PLEXL process description language. The PLEXL language is used to explicitly define the domain of the Plexsys Planning System and configure the Plexsys knowledge base semantic inheritance network and frame structure. Knowledge definition begins with the definition of the constructs which form the context-independent core of the system. Examples of the abstract constructs defined in the Plexsys Planning System are the terms "PROCESS" and "MODEL". Once the

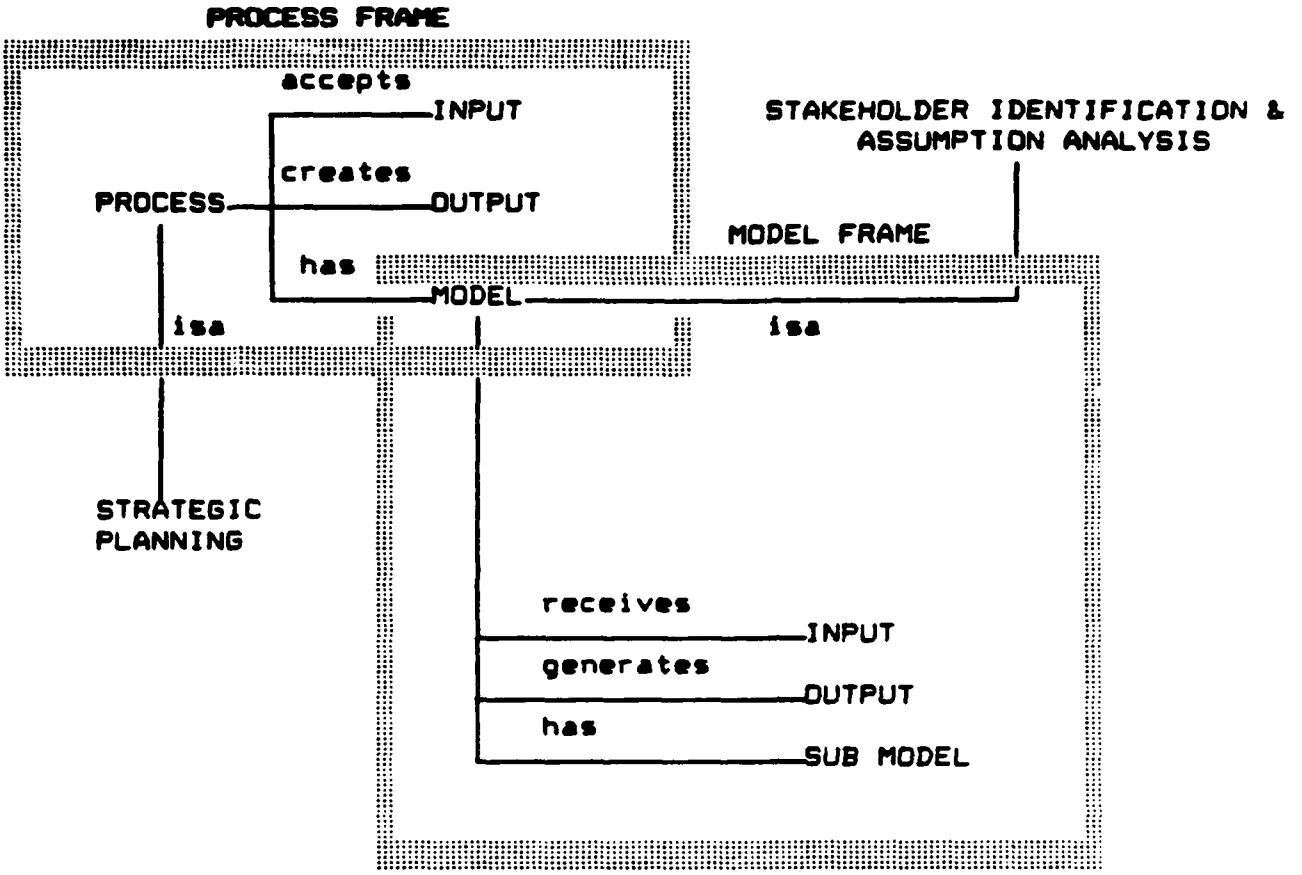


FIGURE 4

SEMANTIC INHERITANCE NETWORK & FRAME REPRESENTATION
WITHIN THE PLEXSYS PLANNING SYSTEM

abstract constructs have been defined, the PLEXL language is used to define increasingly detailed and context-dependent descriptions of these constructs in an effort to fully describe the explicit knowledge of the Plexsys Planning System.

The Knowledge Management System component of the Plexsys Planning System is an integrated workbench of knowledge management tools that allow a knowledge base designer to define the semantic inheritance network and frame knowledge base used to drive the Plexsys Planning System. Four interactive tools are provided.

The Term Editor allows a system designer to identify the abstract terms that form the core constructs for the system. Synonyms for a given term and up to five lines of documentation may also be defined. Figure 5 shows a screen from the Term Editor in which the abstract term "MODEL" has been defined. Each term can be documented with up to a five line description. In addition, up to three synonyms for a term can be defined (e.g., MODELS). A query facility is available that allows the user to query the knowledge base and view abstract terms that have been defined within the system.

The Expression Editor allows a system designer to create expressions using the terms identified with the Term Editor. This provides a user-friendly, interactive tool for defining the semantic inheritance network and frame knowledge representation for the abstract constructs within the system domain. Figure 6 presents a series of screens from the Expression Editor.

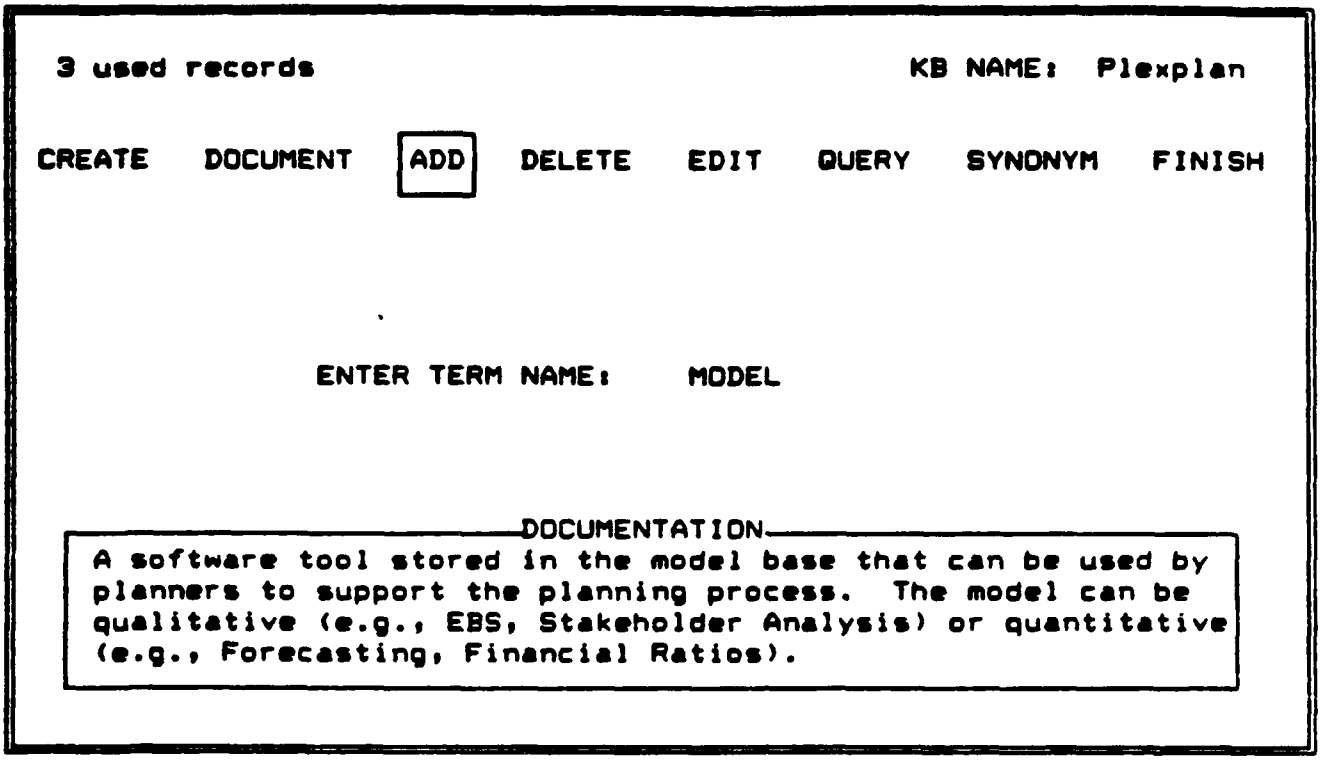


Figure 5

Plexsys Term Editor

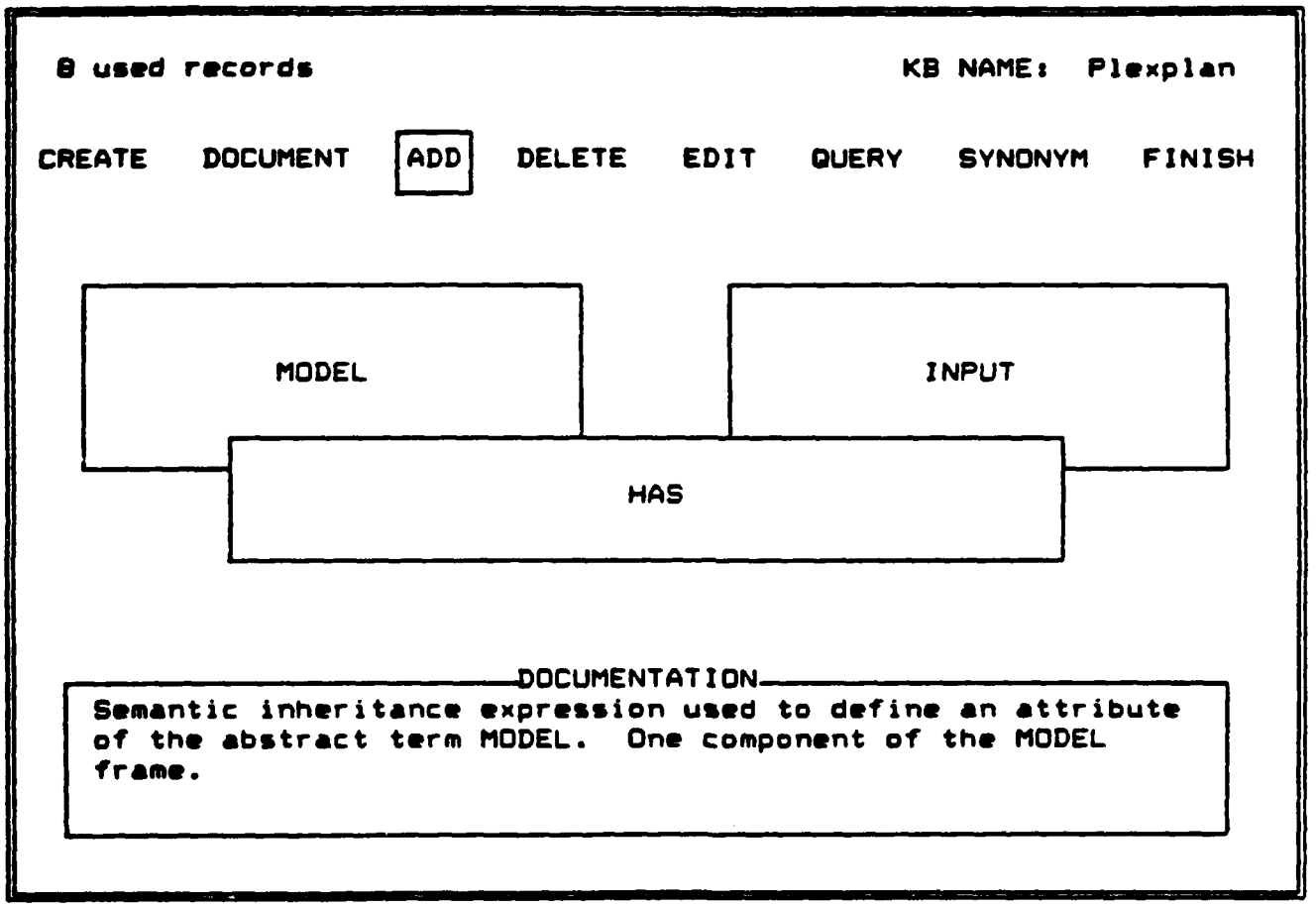


Figure 6

Plexsys Expression Editor
Screen (a)

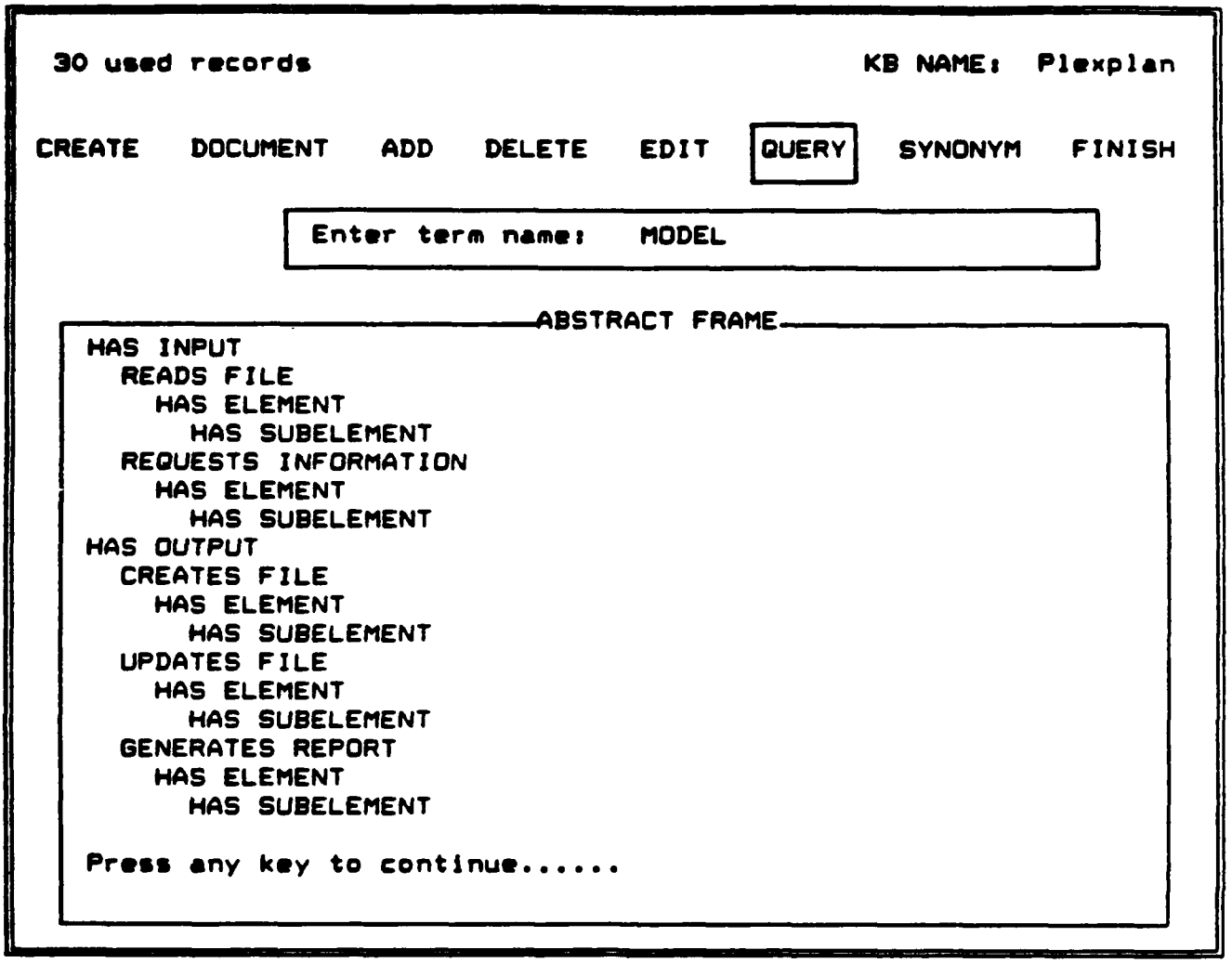


Figure 6

Plexsys Expression Editor
Screen (b)

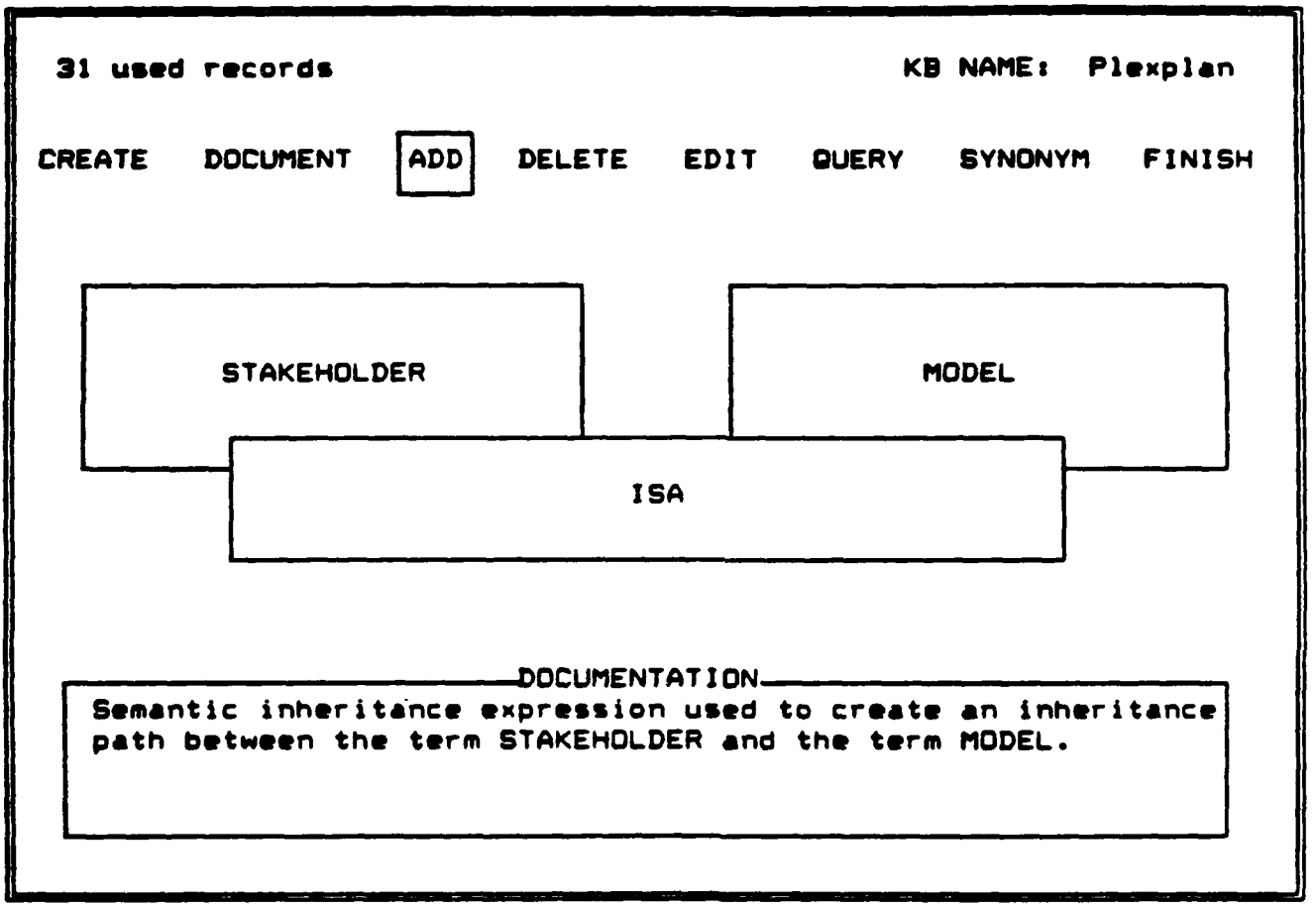


Figure 6

Plexsys Expression Editor
Screen (c)

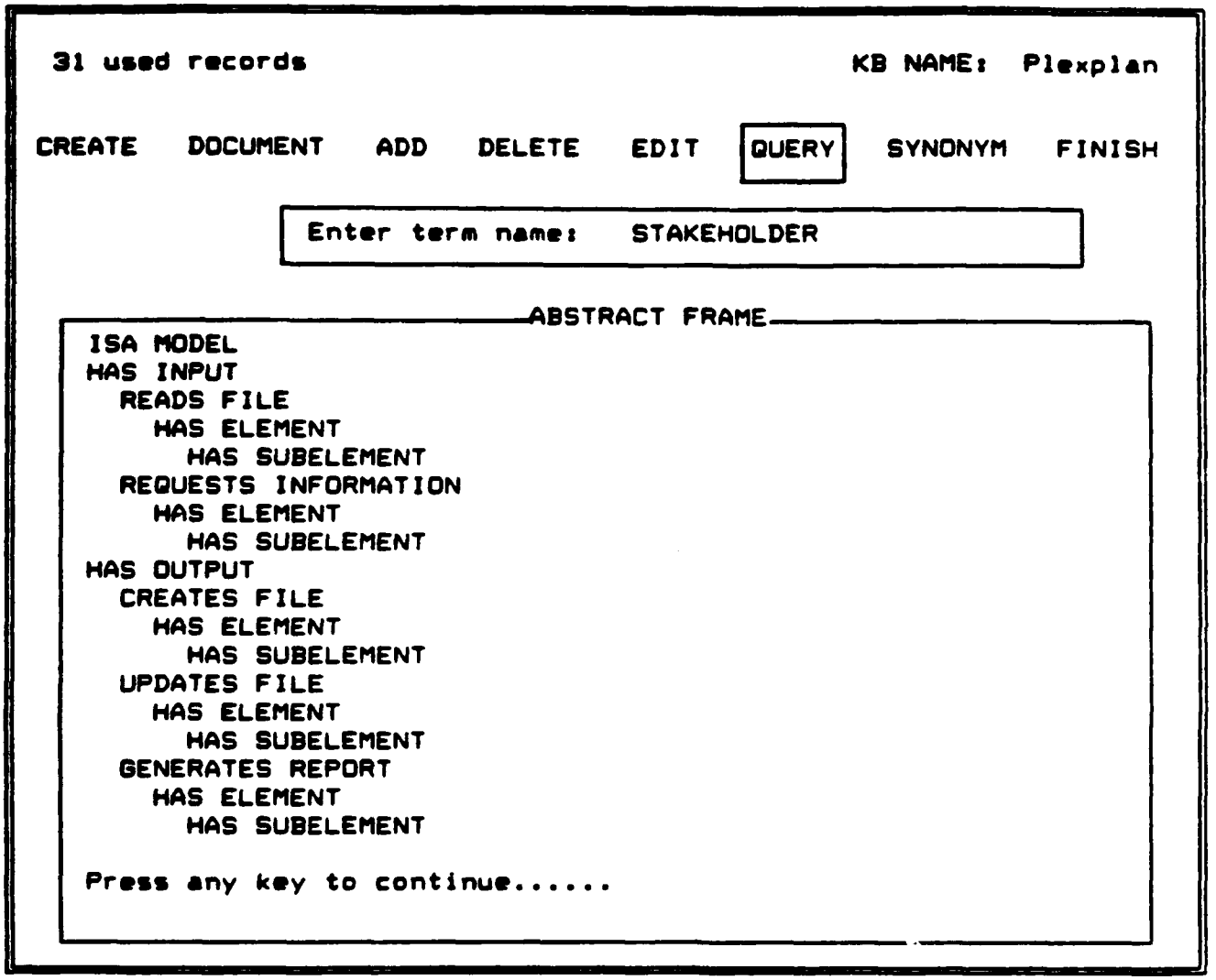


Figure 6

Plexsys Expression Editor
Screen (d)

Screen (a) demonstrates the creation of one expression for the term "MODEL" . The user is prompted to first define a term in window 1 (MODEL), then to define a term in window 2 (INPUT) and finally to enter a relationship term for the two terms in window 3 (HAS). As a result, the expression "MODEL HAS INPUT" is defined and becomes a part of the abstract frame for the term MODEL. Once again a query facility is available that allows the user to query the system and view the abstract frames that have been created and stored in the knowledge base. Screen (b) shows a portion of the abstract frame for "MODEL".

The inheritance properties of the system are demonstrated in screens (c) and (d). Screen (c) shows the definition of the expression "STAKEHOLDER ISA MODEL". Once defined, the term STAKEHOLDER automatically inherits the attributes of the term MODEL as shown in screen (d).

Once the abstract semantic inheritance network and frame structure has been defined, the Median Editor is used to provide detailed definitions of the semantic inheritance network and frame representation within the knowledge base. Increasing levels of context-dependent frames are defined until the planning and decision process representation is one step from the definition of specific strategic planning and decision process and model instances (attribute values).

In the detailed frame description, the knowledge base designer supplies the specific inputs, outputs and sub-models for the Stakeholder Identification & Analysis Model in response to system prompts that are generated by the MODEL frame. Figure 7 shows the

detailed frame for the Stakeholder Analysis model, defined using the Median Editor.

Once the knowledge base is defined on a detailed level, the Knowledge Management system is ready to store planning and decision process information generated by planners using planning and decision models (e.g., Stakeholder Identification and Assumption Analysis; Electronic Brainstorming). This is accomplished through the instantiation of the detailed frames created using the Median Editor. This instantiation process creates instance level frames that represent specific values of the attributes and objects defined thus far in the knowledge base.

The instance level frame is defined and then linked to the appropriate detailed model and process frames. For example, the instance level frame containing information on stakeholders and assumptions generated by the group of planners attempting to decide whether to enter the 32-bit microcomputer market could be linked to the STAKEHOLDER model frame by the expression "32BITSTAKEHOLDER isa STAKEHOLDER" (model). Through the inheritance properties of the network this also implies that "32BITSTAKEHOLDER isa MODEL". The 32BITSTAKEHOLDER information is also linked to the 32BIT MARKET PLANNING process frame.

The storage of planning information in the knowledge base can be accomplished in one of two ways. Batch knowledge base update tools have been developed that automatically create an instance level frame that represents the specific inputs and outputs generated by strategic

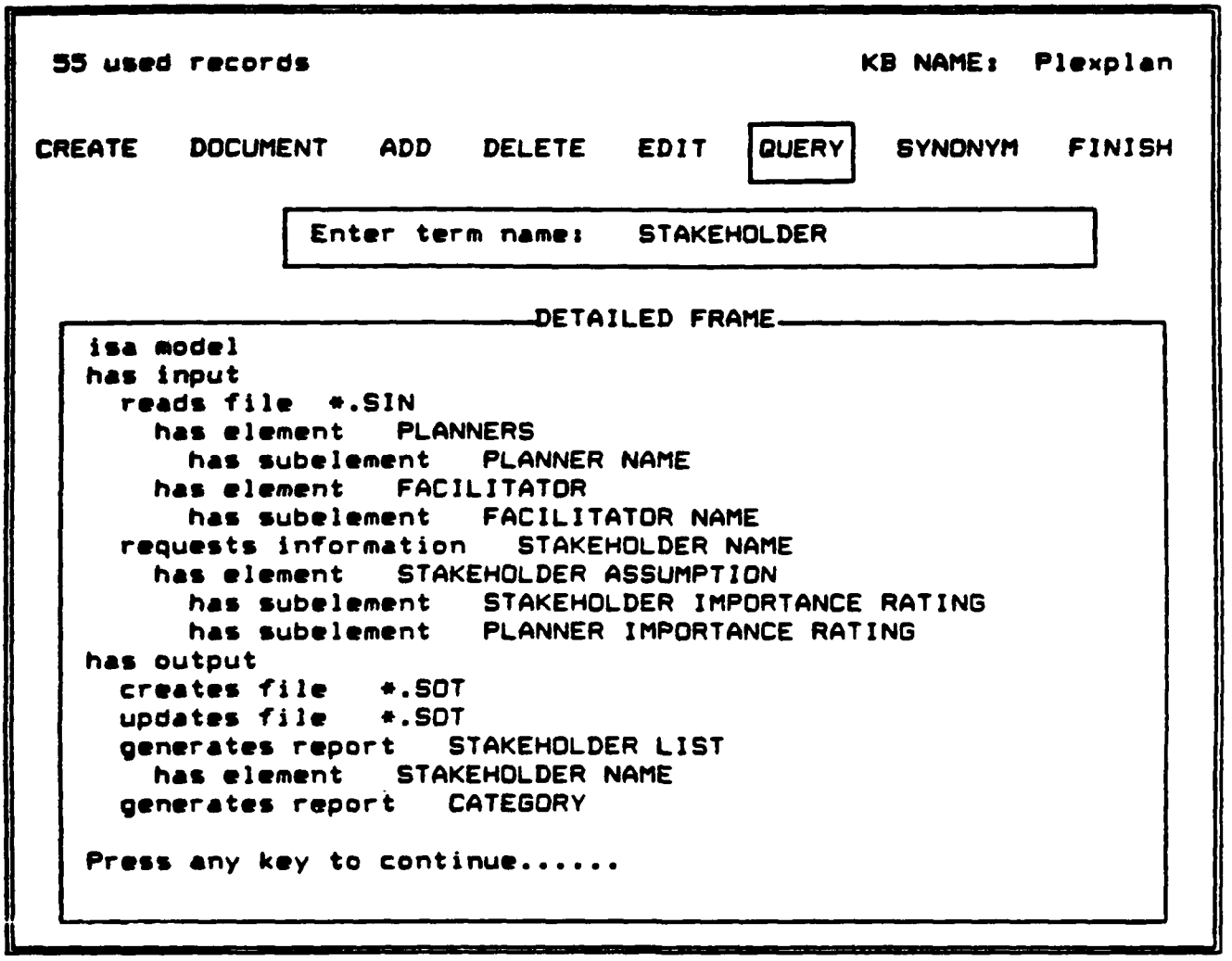


Figure 7

Plexsys Median Editor

planners and decision-makers using the Plexsys planning models. In this case, update of the knowledge base is transparent to the user. An interactive knowledge management tool, the Instance Editor, can also be used to update the knowledge base with specific planning information. Figure 8 presents a screen showing the 32BIT instance frame linked to the STAKEHOLDER model frame.

Screen (a) demonstrates the definition and storage of one stakeholder, IBM, as an instance of the term "STAKEHOLDER NAMES" which is one element of the STAKEHOLDER model. The term "STAKEHOLDER NAMES" has a sub-element "STAKEHOLDER ASSUMPTION". Two assumptions have been defined "ASSUMPTION 1" and "ASSUMPTION 2". These two assumptions are further sub-divided into three parts, "STAKEHOLDER IMPORTANCE RATING (value = 10 and 5 respectively) and "PLANNER IMPORTANCE RATING" (values = 10 and 2 respectively). The actual assumptions are too large to be stored as a term. Instead they are stored as a document linked to the term. [See screen (b).]

In summary, the Knowledge Management System includes four interactive knowledge base editors that allow a knowledge base designer to define semantic inheritance network and frame knowledge representations on three levels: (1) abstract, (2) detailed and (3) instance (value). Instance level frames are defined using batch or interactive knowledge base management tools. (It is important to note that the screens used to demonstrate the Knowledge Management tools are illustrative in nature and do not represent actual information currently stored in the knowledge base).

```
98 used records                                    KB NAME:  Plexplan

CREATE   DOCUMENT   ADD   DELETE   EDIT   [QUERY]   SYNONYM   FINISH

        Enter term name:   32bIT STAKEHOLDER

                          INSTANCE FRAME
isa model
has input
  reads file  *.sin   32BIT.SIN
    has element   planners
      has subelement   planner name   PLANNER1
                                      PLANNER2
    has element   facilitator
      has subelement   facilitator name   APPLEGATE
  requests information   stakeholder name   IBM
    has element   stakeholder assumption   ASSUMPTION #1
      has subelement   stakeholder importance rating   10
      has subelement   planner importance rating   10
      has subelement   certainty rating   4
    has element   stakeholder assumption   ASSUMPTION #2
      has subelement   stakeholder importance rating   5
      has subelement   planner importance rating   2
      has subelement   certainty rating   9
  requests information   stakeholder name   PC DIVISION
    has element   stakeholder assumption   ASSUMPTION #1
      has subelement   stakeholder importance rating   8
      has subelement   planner importance rating   5
      has subelement   certainty rating   4
has output
  creates file   *.sot
  updates file   *.sot
  generates report   stakeholder list
    has element   stakeholder name
  generates report   category

Press any key to continue......
```

Figure 8

Plexsys Instance Editor
Screen (a)

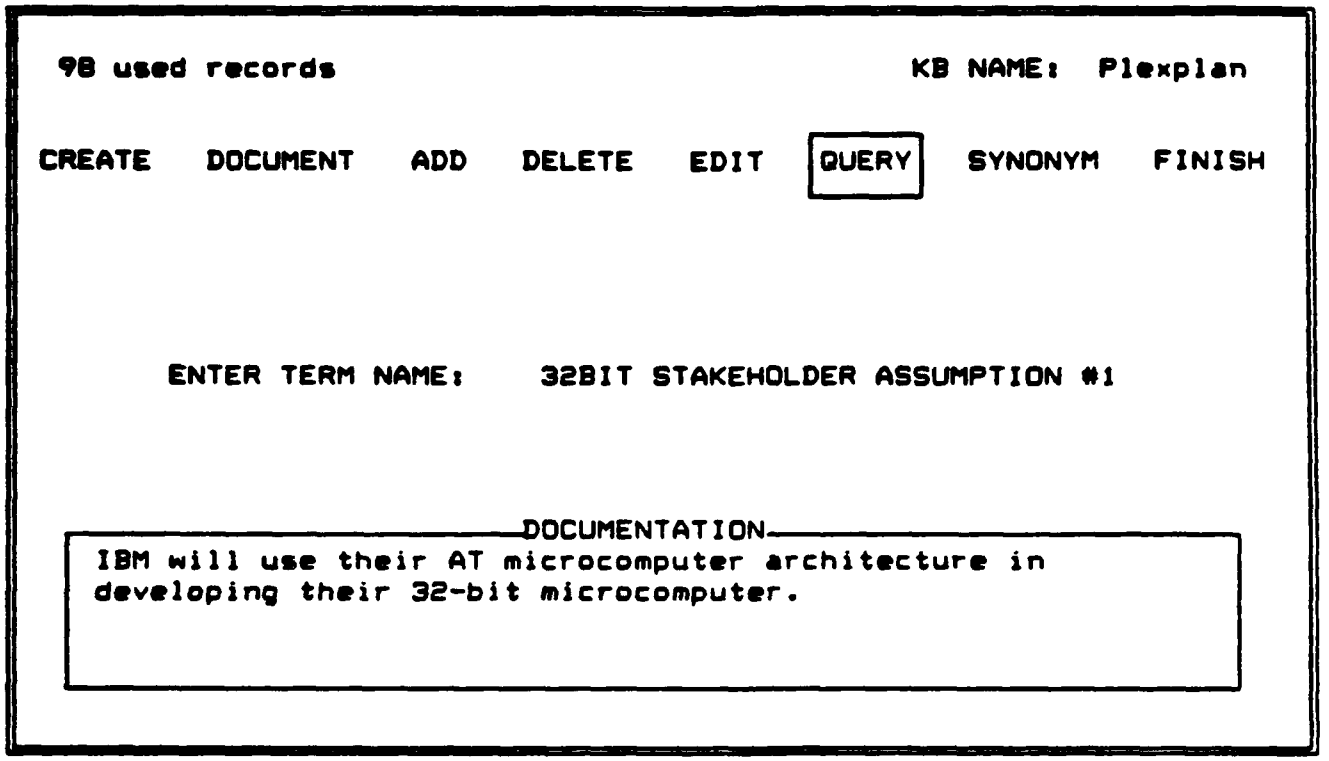


Figure 8

Plexsys Instance Editor Screen (b)

A wide variety of other Knowledge Management tools are available that have not been discussed in this paper. These tools assist with the maintenance of the knowledge base system, analysis of the consistency and integrity of the knowledge base and knowledge base security. Network management tools are also available to control the use of the knowledge base and planning/decision models in the group decision support environment of the MIS Planning and Decision Laboratory.

4.0 SUMMARY AND CONCLUSIONS

In summary, the concept of automated support for decision-making in organizations has undergone significant development since it was first proposed in the early 1970s. Research in this area has provided a framework to guide the design and implementation of DSS within organizations. Studies on the impact of DSS are beginning to provide evidence of the influences of these systems on organizational decision-making and structure. Aided by advances in information technology, DSS are now widely used throughout organizations.

The next generation of systems to support organizational decision-making has been proposed. These systems will expand the concept of decision support to include support of (1) unstructured, complex decision processes in addition to structured, decision-specific tasks; (2) deliberation and judgement in addition to quantitative data analysis; (3) individual and group decision-making; and (4) improved effectiveness of organizational decision-making through the use of knowledge-based and expert systems approaches.

Changes in the design of DSS are necessary to accomplish the advances described above. Knowledge management and expert system techniques are required to enable the system to support complex, unstructured decision processes and effective decision-making. Process management is required in addition to the traditional data management, dialogue management and model management to reflect the increased emphasis on support of decision processes. Network management is required to enable group decision support consistent with the group nature of upper level management decisions. Finally, a flexible dialogue management system is required to permit multiple user interface, access and presentation modes.

This paper has explored the technical feasibility of future automated systems for direct support for deliberation and judgement. The Plexsys Planning System was discussed as an example of a knowledge-based DSS architecture that integrates process management, model management and data management in a group decision-making and planning environment. The system is designed to support planning and decision-making at all levels of the organization. Special emphasis, however, is placed on the support of deliberation and judgement processes at the upper levels of organizations.

Planning and decision processes are represented through the use of a semantic inheritance network and frame knowledge representation. A Knowledge Management system has been developed that provides a workbench of software tools for defining and managing the planning and decision process representations. Planning and decision models (both analytic and qualitative) are represented in the semantic inheritance

network and frame knowledge base. Data from internal and external organizational database management systems are accessed by the Process Management system to support the decision-making and planning process.

The system has been developed to support both individuals and groups with special emphasis on group decision support. A Network Management system provides a set of software tools to assist in managing the group nature of the system.

The Plexsys Planning System has been implemented in the MIS Planning and Decision Laboratory and has been used by over 100 planners to support strategic planning sessions. Evaluation of the use of the system indicates that the deliberation and judgement process and the qualitative models used to support it can be accurately represented and stored in increasing levels of detail using the Plexsys Knowledge Management System tools and the PLEXL process description language. Information gathered from the planners using these models can then be represented and stored in the system as an instance of process and model frames. This information can be readily retrieved for future analysis. Planners report high levels of satisfaction with the use of the system (Applegate, 1986; Applegate, Nunamaker and Konsynski, 1986; Nunamaker, Applegate and Konsynski, 1987).

Future research efforts are directed toward the integration of the Plexsys Planning System with the Plexsys Intelligent Information System Development Environment (Konsynski and Nunamaker, 1982; Konsynski, Kotteman, Nunamaker and Stott, 1984; McIntyre, Konsynski and Nunamaker, 1986). In addition, the Plexsys Planning System will be expanded to provide an enhanced set of planning and decision models and to enable

the more accurate and flexible storage and management of normative planning and decision rules. The changes to the system necessary to permit its use within organizations are also being evaluated.

BIBLIOGRAPHY

Ackoff, R.L., (1970). A Concept of Corporate Planning. NY:McGraw Hill.

Applegate, L.M. (1986). Idea Management in Organization Planning. Unpublished doctoral dissertation, University of Arizona.

Applegate, L.M., Klein, G., Konsynski, B.R. and Nunamaker, J.F. (1985). Model management systems: Proposed representations and future designs. Proceedings of the Sixth Annual International Conference on Information Systems. Indianapolis.

Applegate, L.M., Konsynski, B.R. and Nunamaker, J.F. (1986). Model management systems: Designs for decision support. Decision Support Systems. 2(1):81.

Applegate, L.M., Konsynski, B.R. and Nunamaker, J.F. (1986). A group decision support system for idea generation and issue analysis in organizational planning. Proceedings of the Conference on Computer-Supported Cooperative Work. Austin.

Applegate, L.M., Konsynski, B.R. and Nunamaker, J.F. (1987). Knowledge management in organization planning. Journal of Management Information Systems (forthcoming - Spring).

Applegate, L.M., Mason, R.O. and Thorpe, D.P. (1986). Design of a management support system for hospital strategic planning. Journal of Medical Systems. 10(1):79-95.

Argyris, C. (1970). Intervention Theory and Method. Reading, Mass: Addison-Wesley.

Benbasat, I. (1984). An analysis of research methodologies. The Information Systems Research Challenge. McFarlan, F.W. (ed). Cambridge: Harvard Business School Press.

Blanning, R. (1982). A relational framework for model management in decision support systems. DSS-82 Transactions.

Bonczek, R.H., Holsapple, C.W. and Whinston, A.B. (1981). A generalized decision support system using predicate calculus and network data base management. Operations Research. 29(2):263.

Brachman, R.J. and Levesque, H.J. (1985). Readings in Knowledge Representation. Los Altos: Morgan Kauffman.

Bui, T. and Jarke, M. (1984). A DSS for cooperative multiple criteria group decision making. Proceedings of the Fifth International Conference on Information Systems.

Data Decisions (1983). Micros at big firms - a survey. Datamation. 29:126-141.

DeSanctis, G. and Gallupe, B. (1985). Group decision support systems: A new frontier. Data Base. Winter.

Dhar, V. (1986). On the plausibility and scope of expert systems in management. Proceedings of the 19th Hawaii International Conference on System Sciences.

Dolk, D.R. and Konsynski, B.R. (1984). Knowledge representation for model management systems. IEEE Transactions on Software Engineering. SE-10(6):619.

Elam, J., Henderson, J., and Miller L.W. (1980). Model management systems: an approach to decision support in complex organizations. Proceedings of the First Conference on Information Systems.

Fuglseth, A.M. and Stabell, C.B. (1985). Capture, representation and diagnosis of user information perception. Knowledge Representation for Decision Support Systems. (eds) Methlie and Sprague. Elsevier Publ.

Gallupe, R.B. (1986). Experimental Research into Group Decision Support Systems: Practical Issues and Problems. Proceedings of the Nineteenth Hawaii International Conference on System Sciences. 1986.

Gerrity, T.P. (1971). The design of man-machine decision systems: An application to portfolio management. Sloan Management Review. 12(2):59.

Gershefski, G.W. (1969). Corporate planning models - the state of the art. Managerial Planning. 18:31-35.

Gibson, C.F. (1975). A methodology for implementation research. Implementing Operations Research/Management Science. Schultz, R.L. and Slevin, D.P. (eds). NY: Elsevier.

Gorry, G.A. and Scott Morton, J.S. (1971). A framework for management information systems. Sloan Management Review. 13(1):55.

Gray, P. (1981). The SMU decision room project. DSS-81 Transactions.

Hiltz S.R. and Turoff, M. (1981). The evolution of user behavior in a computerized conferencing system. Communications of the ACM. 24(11):739.

Huber, G. P. (1982). Group decision support systems as aids in the use of structured group management techniques. DSS-82 Transactions.

Huber, G.P. (1984). Issues in the design of group decision support systems. Management Information Systems Quarterly, September.

Isenberg, D.J. (1984). How senior managers think. Harvard Business Review. November-December.

Keen, P.W. and Gambino, T.J. (1983). Building a decision support system: The mythical man-month revisted. Building Decision Support Systems. Bennett, J.L. (ed). Reading, Mass: Addison-Wesley.

Kein, R.T. and Philippaks, A.S. (1985). Decision support systems in practice: profile on management and professional workstations. DSS-85 Transactions.

Konsynski, B.R. and Dolk, D. (1982). Knowledge abstractions in model management. DSS-82 Transactions.

Konsynski, B.R., Kotteman, J., Nunamaker, J.F. and Stott, J. (1984). Plexsys-84: An integrated development environment for information systems. Journal of Management Information Systems. 1(3):64.

Konsynski, B.R. and Nunamaker, J.F. (1982). Plexsys: A system development system. Advanced System Development and Feasibility Techniques. (eds) Cougar, Colter, Knapp. NY:John Wiley and Sons.

Kraemer, K.L. and King, J.L., (1984). Computer-based systems for group decision support. Proceedings of the National Academy of Management Annual Conference. 1984.

Levesque, H.J. and Barachman, R.J. (1985) A fundamental tradeoff in knowledge representation and reasoning. Readings in Knowledge Representation. Barachman and Levesque (eds).

Little, J.D.C. (1986) Research opportunities in the decision and management sciences. Management Science. 32(1):1-13.

McIntyre, S.C., Nunamaker, J.F. and Konsynski, B.R. (1986). Automating planning environments: Knowledge integration and model scripting. Journal of Management Information Systems.

Minsky, M. (1975). A framework for representing knowledge. The Psychology of Computer Vision. Whinston, P. (ed). NY:McGraw Hill. reprinted in Readings in Knowledge Representation. Barachman and Levesque (1985).

Mintzberg, H. (1973). The Nature of Managerial Work. NY:Harper and Row.

Naylor, T.H. (1976). Conceptual framework for corporate modeling and the results of a survey of current practices. Operations Research Quarterly. 27:671-682.

Naylor, T.H. (1982). Strategic planning models. Managerial Planning. 30:3-11.

Naylor, T.H. and Schauland, H. (1976). A survey of users of corporate planning models. Management Science. 22:927-937.

Nunamaker, J.F., and Applegate, L.M. (1985). Information System Planning for the Amphi School District. Technical Working Paper Series, University of Arizona.

Nunamaker, J.F., Applegate, L.M. and Konsynski, B.R. (1987). Facilitating group creativity: Experience with a group decision support system. Journal of Management Information Systems. (forthcoming - Spring).

Nunamaker, J.F. and Konsynski, B.R. (1979). Information System Planning for the Library of Canada. Technical Working Paper Series, University of Arizona.

Nunamaker, J.F., Swenson, D.E. and Whinston, A.P. (1973). Specifications for the development of a generalized database planning system. AFIPS Conference Proceedings.

Phillips. L.D. (1984). A theory of requisite decision models. ACTA Psychologica. 56:29.

Quillian, M.R. (1968). Semantic memory. Semantic Information Processing. (ed) Minsky. Cambridge:MIT Press.

Scott Morton, M.S. (1971). Management Decision Systems: Computer Based Support for Decision Making. Cambridge: Harvard Business School Division of Research.

Sprague, R. (1980). A framework for the development of DSS. MIS Quarterly. 4(4), December.

Stabell, C.B. (1979). Decision research: Description and diagnosis of decision-making in organizations. Institute for Information Systems Research, Norwegian School of Economics and Business Administration Working Paper #A-79.006.

Stabell, C.B. (1983). A decision-oriented approach to building DSS. Building Decision Support Systems. Bennett, J.L. (ed). Reading, Mass: Addison-Wesley.

Susman, G.I and Evered, R.D. (1978). An Assessment of the Scientific Merits of Action Research. Administrative Science Quarterly. 23:582-603.

Will, H.J. (1975). Model management systems. Information Systems Structure and Organizations. (eds) Grochla and Syperski, Berlin: Walter de Gruyter.

THOUGHTS ON INFORMATION ASSET MANAGEMENT

Dan S. Appleton

D. Appleton Co.

Contributed by Dan S. Appleton

THOUGHTS ON
Information Asset Management

By Daniel S. Appleton

In 1972 John Dearden published "MIS is a Mirage," an article that put him at odds with much of the data processing community.

> "The notion that a company can and ought to have an expert (or a group of experts) create for it a single, completely integrated super-system - an "MIS" - to help it govern every aspect of its activity is absurd." [DEAR 72]

Dearden also pointed out that:

> "It is difficult even to describe the MIS in a satisfactory way, because this conceptual entity is embedded in a mish-mash of fuzzy thinking and incomprehensible jargon. It is nearly impossible to obtain any agreement on how MIS problems are to be analyzed, what shape their solutions might take, or how these solutions are to be implemented." [DEAR 72]

What has happened since 1972? If you reread Dearden's article, you will probably think that nothing has happened. It seems just as true today as it did in 1972.

It is not. The problem that Dearden was addressing - that of creating "a single, completely integrated super-system" - was a spurious problem. What made it seem real was the engineering life cycle model that dominated Dearden's thinking about information resource management (IRM). This model I call the "job management" model. It assumes that IRM is accomplished using the traditional approach: design, produce, implement, and maintain. And, it led Dearden to describe a situation wherein the MIS was a single system, designed, produced, implemented, and maintained by experts.

The job-based engineering life cycle model still dominates IRM today. While we no longer think in terms of a single super system designed from the top down and implemented from the bottom up, we do think about planning a single enterprise-wide super systems from the bottom up. The traditional engineering life cycle model is alive and well in IRM circles.

But, because it is not economical and because it does not create an adaptive environment, that life cycle model is a dinosaur. It is being displaced by a new life cycle model. This model I call the "asset-based engineering life cycle" model. This model assumes that IRM is accomplished through the development and continual reuse of information assets.

The trend toward asset management is the megatrend in enterprise-wide IRM. It is having a dramatic effect on how information managers plan, organize, staff, measure performance, set standards, define and control work, and, ultimately, produce results. This paper describes the megatrend toward asset-based IRM.

CAN INFORMATION BE MANAGED?

There is an old saying: "Life is what goes on while you are planning something else." The same could be said for IRM. Information is being managed, today. Each business enterprise has a specific management methodology for enterprise-wide IRM.

So what is all the fuss about IRM and total MISs? Well, it all has to do with the notion of "on purpose." Do businesses manage information "on purpose," or "by accident." IRM "by accident" is as much a management methodology as IRM "on purpose."

In most businesses the IRM advocates would like to change the management methodology from "by accident" to "on purpose." That shift alone would be a major victory. Probably 80 percent of today's information management methodologies fall into the "by accident" category. Why? Because most business executives still believe with Dearden that:

1. "The true MIS expert does not and cannot exist."

2. "Coordinated systems for functional areas can be developed without a total systems approach."

3. "'The systems approach' is merely an elaborate phrase for 'good management.'"

4. "Centralizing the control of a company's information systems in a staff group creates problems that are insoluble; therefore, it is simply not feasible."

Obviously, gaining a shift away from this style of IRM toward the "on purpose" style has a dramatic impact on the whole enterprise culture. Such shifts are happening, for the reasons proposed in Alvin Toffler's The Third Wave and John Naisbett's Megatrends. Enterprise cultures are beginning to transform general management into a style that favors IRM as a conscious act.

This presents a new problem. What form of "on purpose" IRM do they want?

Regardless of who is managing information in an enterprise - a centralized DP function, a federation of information professionals, or a participative group of business professionals - IRM has four fundamental objectives:

1. Increase the availability of information
2. Increase the quality and reliability of a information
3. Reduce the cost per unit of information
4. Increase the visibility and control provided by information

These objectives never change. All that changes are the approaches to achieving them.

In 1983, F. Warren McFarlane and James L. McKenney published an excellent book called Corporate Information Systems Management, The Issues Facing Senior Executives. In Chapter 1 they say, "Virtually all major, currently accepted conceptual frameworks for thinking about how to manage in this [information systems] field have been developed since 1971." [MCFA 83] Not only do I believe this to be untrue, I also believe it to be a dangerous thought. It is this idea which has led to the evolution of what someone once called "the framework of the month club," and to a general confusion about what is important in managing information.

The basic management model for IRM is not driven by technology (which would account for the illusion of newness). It is driven by economics. It is driven by the age-old, drab problem of supply and demand. This is why we have so many good paradigms available to us for improving IRM. Virtually all major, currently accepted conceptual frameworks for thinking about how to manage in this field were developed prior to 1971. They were probably even known to Aristotle.

THE MANUFACTURING PARADIGM

There are plenty of models that can be used in evolving the management principles and philosophies of IRM. These models have been studied by academicians and philosophers for a hundred years. [APPL 81] They have been widely used by all kinds of businessmen. The most refined and studied model, and the model most appropriate for developing effective theories of information resource management, is the manufacturing enterprise.

By the manufacturing enterprise, I do not mean simply shop management. Manufacturing includes all of the marketing, program management, engineering, production, planning, finance, and field support functions generally involved in a total manufacturing enterprise. These same manufacturing functions are contained within the typical data processing or information resource management organization. It is literally a business within a business.

Manufacturing organizations can be categorized in one of four stages of evolution. (Figure 1) Those organizations which are controlled by their marketplace are generally called job shops. They are also "pure service" organizations. Their assets are primarily fixed assets (machines, equipment, facilities, etc.).

Productivity in a job shop is achieved by leveraging expenses against market demand. Because in a free market, demand is fairly arbitrary, and there is no concept of a "product" in a job shop (i.e., it provides a service, not a product). Its control of expenses is based upon the requirements of individual customer orders. Nothing is accomplished in the organization until an order is received. Once an order is received, the order is engineered, material is procured, and production resources are specifically committed to it. Job shops are notoriously inefficient and unpredictable in terms of production lead times and costs, and they do not get the best margins.

In a job shop, everything happens in series. (Figure 2-A) Each order is treated uniquely, and there is very little effort towards standardization. The general attitude is that the customer is always right, and we must do what he wants us to do or lose the order. There is little or no "value added" inventory. Engineers (and customers, as well) spend a lot of time in the manufacturing environment re-engineering and controlling individual orders as they move through the production cycle, and each individual order maintains its identity as a unique experience - even after completion.

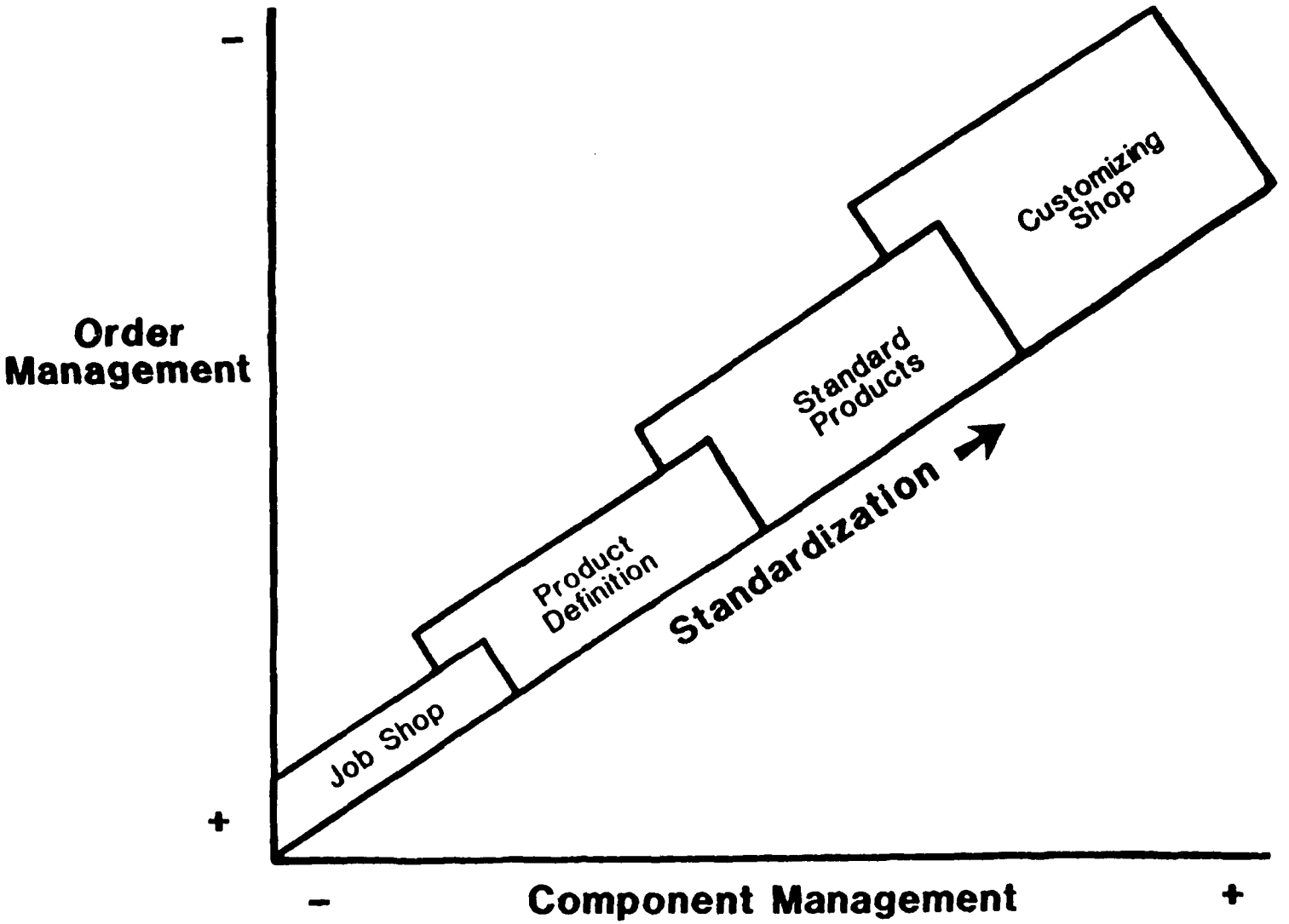

Order Management
-
+
Job Shop
Product Definition
Standard Products
Customizing Shop
Standardization
-
Component Management
+

Figure 1

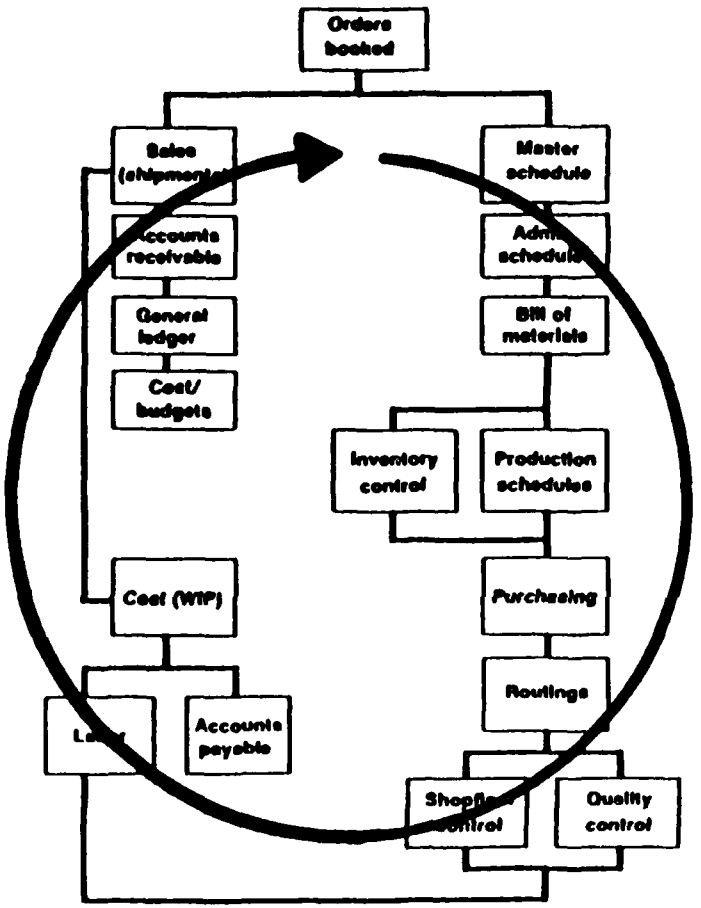


Figure 2A

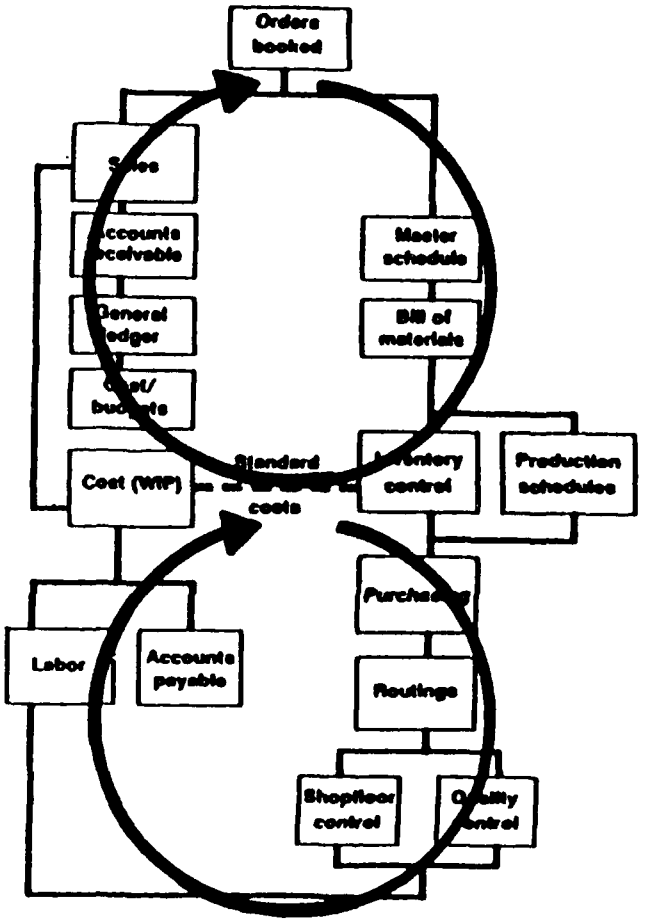


Figure 2B

Under this definition, job shops are a sign of slovenly management. They are not driven by the marketplace (though they claim to be), nor are they driven by a basic product philosophy. In fact, they do not even produce a product per-se. Products are brought to a market; they are not developed by a company for a specific customer. That is a service.

Most manufacturing organizations start off as job shops, but they do not have to. Generally their job shop phase is controlled by their financial management strategy (i.e., bootstrapping vs. external capital infusion). But job shops will always remain job shops if they do not begin to define what their products are. Generally, this is done through standardization programs that analyze individual customer orders, looking for areas of potential standardization.

Product definition is accomplished through standardization. It is absolutely necessary to take this step before value-added assets can be developed and subsequently leveraged to reduce product development cost and leadtime and improve product quality and predictability.

A typical scenario of product definition occurred at one of my business alma maters - Byron Jackson Pump (BJP). In its job shop days, BJP engineered all pumps for each customer order. After many years, someone asked, "How many impellers have we designed?" The answer was basically, one for each customer order. We had drawings for thousands of impellers. An engineering team was asked to look into impeller design. It concluded that there were basically three standard impeller configurations from which the thousands could be customized. The same standardization concept was undertaken for wear rings, volutes, cases, etc. Once the product structure was defined, BJP could "pre-make" parts, store them in semi-finished goods inventory, and use them to reduce product cost, lead time, etc. They did not necessarily limit customer requirements, either. But, they did know when "special" customer requirements would increase cost, lead time, etc.

Standardization must be accomplished by the engineering function. This is difficult in many cases because engineers would rather design applications for a customer than build standard product concepts for the business.

The establishment of standard product concepts allows for the identification of specific types of components which can be leveraged through inventory. Inventory is a very specific asset which is created and subsequently leveraged by the business. It is not procured. The business always adds value to inventory.

Only after product definition can an organization move into a process shop or standard production environment. Process shops attempt to completely organize and control the production of products and to optimize return on investment on all company assets.

A standard production environment operates two separate, but interdependent, management structures in parallel. (Figure 2-B) One management structure identifies demand and delivers the product, the other management structure builds the internal assets that are to be leveraged by the first.

Standard production is not the final state of effective management in manufacturing. That state, according to Alvin Toffler in The Third Wave, is an environment where a company's assets are completely optimized and leveraged, and the customer believes that the product was constructed especially to his unique requirements. This nirvana is the most difficult management problem there is. It is what I call a "customizing shop."

A customizing shop can only be constructed after product definition has been accomplished, and after the business has learned how to build to and ship from stock (i.e., build and manage value added assets such as inventory). It can only operate on a foundation of standard production and with a control philosophy that is consistent with standard production. It cannot return to the embryonic job shop and pure service philosophy. A custom shop must to go through the product definition phase in order to define what can be delivered to the market. It must also go through the standard production phase in order to build the management capability that creates and uses value-added assets such as inventory.

The custom shop has a hybrid management philosophy wherein both the sequential job shop management concept and the parallel standard production shop management process cohabitate. (Figure 2-C) It is constantly balancing the two concepts based upon the existing mix of market demand and the necessity for building and leveraging various assets.

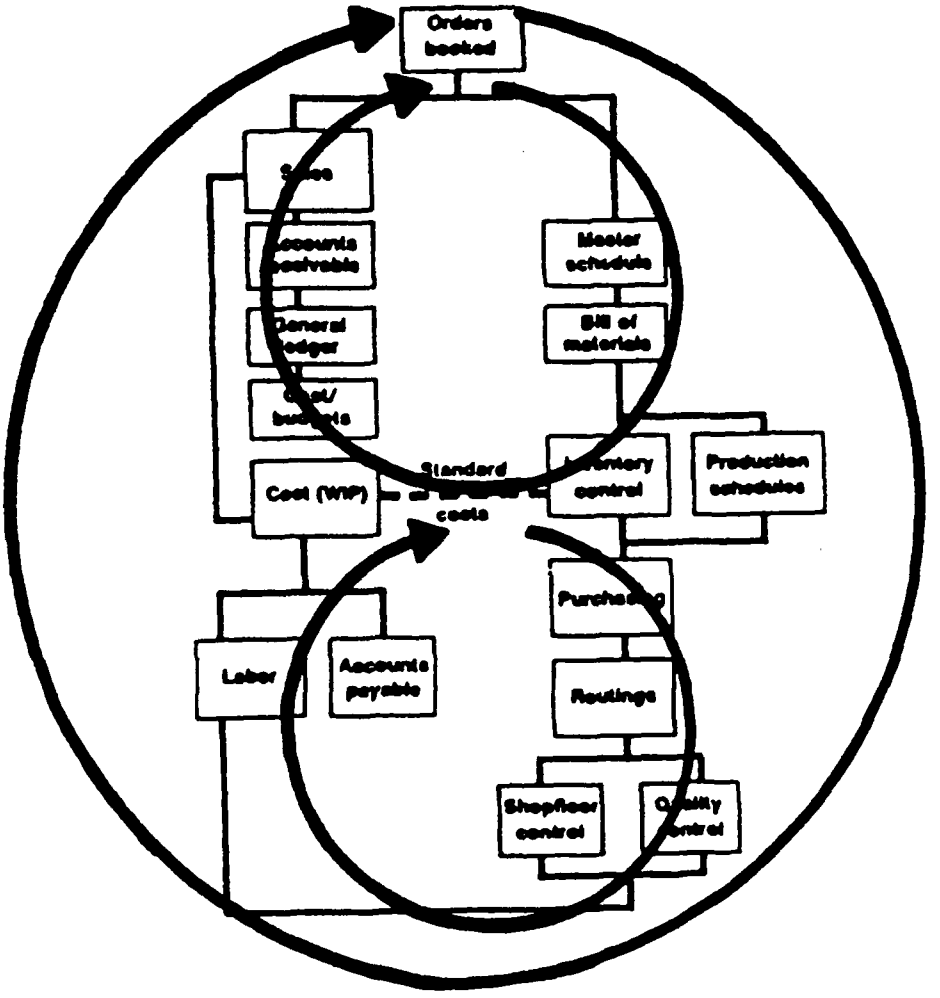


Figure 2C

Within the context of any manufacturing model, the custom shop is indeed the "factory of the future." Its purpose is to respond to a highly dynamic market with an optimally efficient asset management strategy that produces products with optimum financial consequences. It must be responsive to changes in the market as well as changes in product technology. It must be expert at building asset structures internally, which it in turn leverages, in order to gain economies of scale and cumulative volume. It is the ideal model for data processing and information management.

THE TEN JOBS OF A MANAGEMENT METHODOLOGY

I have been using the manufacturing paradigm to characterize the various phases of what I call the "management methodology." Professional managers in any business intuitively understand the notion of a "management methodology." This methodology embodies the "Geist" of their organization. It is articulated through policies, procedures, decision strategies, methods, systems, philosophies, organizational structures, and principles.

Management methodologies are unique to each organization. Sears management methodology is what makes it different from Penney's, Buffums', or Harrod's. If all of the physical assets and people that are the familiar manifestation of Sears were to suddenly disappear, Sears would still exist as a management methodology. That management methodology could construct inventories, buy buildings, employ personnel, define procedures, implement buying programs, etc., from scratch, ultimately manifesting a Sears store. It would not create a Buffums or a Broadway. Nor would the McDonald's management methodology create a Burger King, or Hughes Aircraft Company's management methodology create a Northrop.

Each data processing organization has its own management methodology which distinguishes it from all other data processing organizations. It is at the same time a whole management methodology in and of itself, and a part of the overall management methodology of its parent organization. If it is synchronized with its parent, it is accepted and appreciated. If it is not synchronized, it is ultimately rejected by its host.

The management methodology that is most relevant in characterizing what data processing managers must shoot for is a manufacturing custom shop. However, there is no such concept as a "turnkey" data processing custom shop. It is up to data processing managers to understand and to evolve their existing management methodologies, from the typical job shop methodology, through the various phases of product definition, standard production, and ultimately, to custom production.

Figure 3 depicts the role of the management methodology in any business environment. The driving factor in this figure is, of course, the marketplace. The market drives and to a large extent controls the mission of a business (1), and defines the demand for specific products (2). However, the market must be interpreted based upon a set of structures known to the business management methodology (3). Part of management's job is to define a way of talking about the market and then to map its definition of the market back into distinct structures defining the mission and the products of the organization.

The mission of the organization drives its architecture (4). Very simply, the architecture is a set of homogeneous and consistent rules about the relationships which exist between things in the business environment. These rules are specifically organized toward accomplishing the mission.

The architecture provides a control (5) over the management methodology; however, the management methodology must be capable of adjusting the architecture (6) based upon the needs for short-term, pragmatic responses to changes in the market environment.

The management methodology must produce products which respond to requirements (7) and must extract requirements for products (8). In the job shop, the management methodology basically takes the requirements and produces a specialized solution strategy for them, calling it (euphemistically) a product. This process is reflected in the serial nature of the activities which go on in a job shop after order entry.

After management has evolved its methodology through product definition, the management methodology can begin to create an asset structure (9) which it will in turn leverage to reduce the cost and time and increase the quality and predictability of its products for its markets (10). However, just creating these assets does not ensure they will be employed to optimum benefit. Many manufacturing organizations are reasonably good at creating inventories, but they are terrible at using them effectively.

In a standard production environment, the management methodology produces and builds assets through one mechanism while deploying those assets in the form of products through another mechanism, as reflected in the parallel management strategies shown in Figure 2-B.

Managing the Information Enterprise

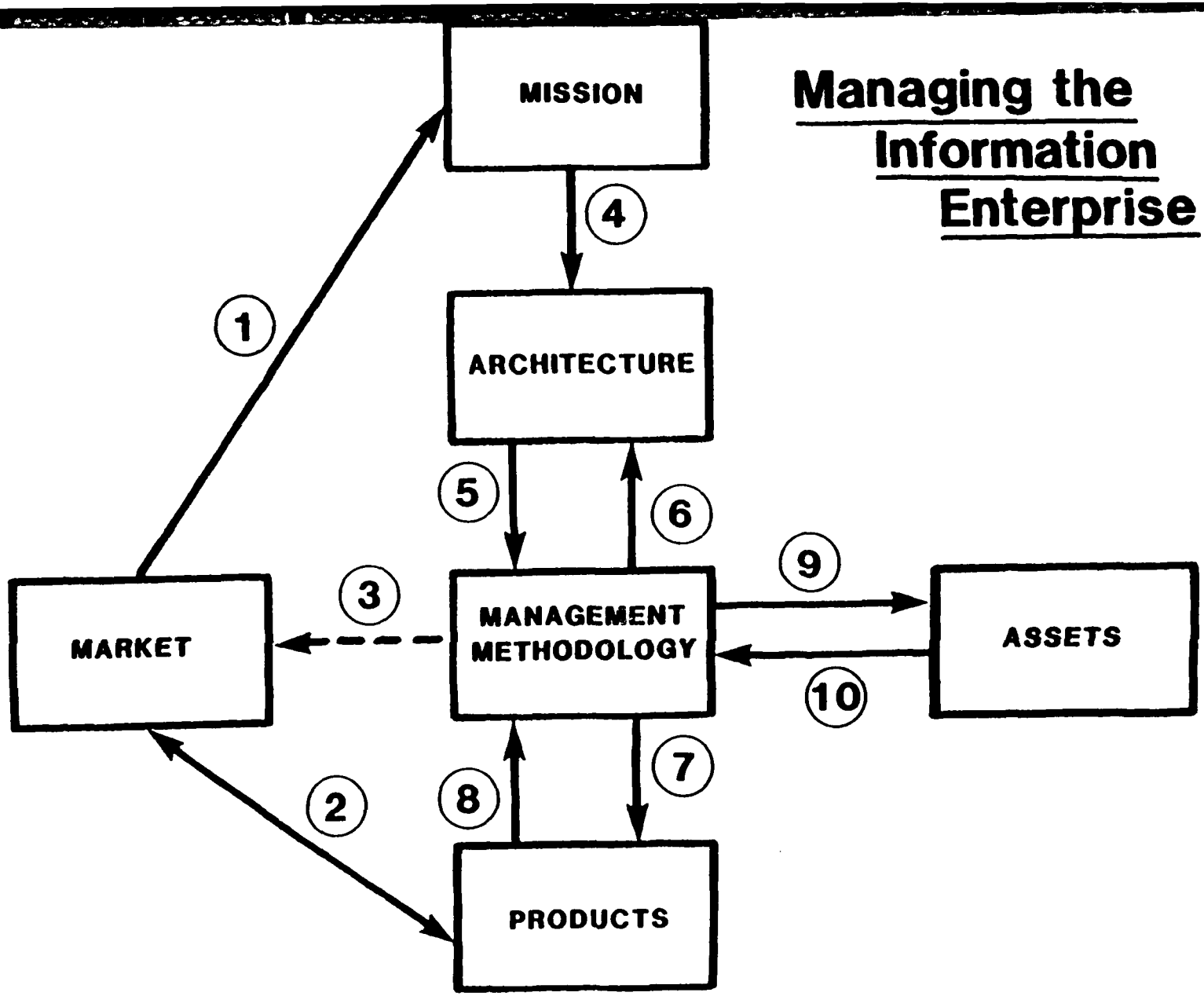


Figure 3

Very few data processing management methodologies have the ability to create an asset structure, much less to use one effectively. The reason they are unable to do this is because they have no notion of product and have not been able to articulate any inventory philosophy relative to their product concepts. Some of them attempt to build and store applications or even program modules, but they generally fall down when it comes to reusing those modules to create products for the customer. One of the reasons for this is that it takes a different kind of money to create assets which can be used in the future than it does to provide services to a customer in the present. The company's money (indirect) is used to create assets which can be leveraged subsequently, while a customer's money (direct) is generally used to build and provide specific services. The former is generally called capital. The latter is called expense.

The management methodology in a custom shop contains the ability to service individual customer requirements and at the same time to build assets, leverage them to speed up product development and delivery, or reduce their cost. Arriving at this level of sophistication in data processing is still a goal. There are no IRM organizations that I know of that have implemented well defined management methodologies for doing so.

THE MATURATION OF MANAGEMENT METHODOLOGY

The management methodology in a job shop is based on the engineering life cycle model shown in Figure 4. All of the steps are executed for each customer order. To a large extent, the productivity of this concept is a function of how fast a system can be designed and built from scratch.

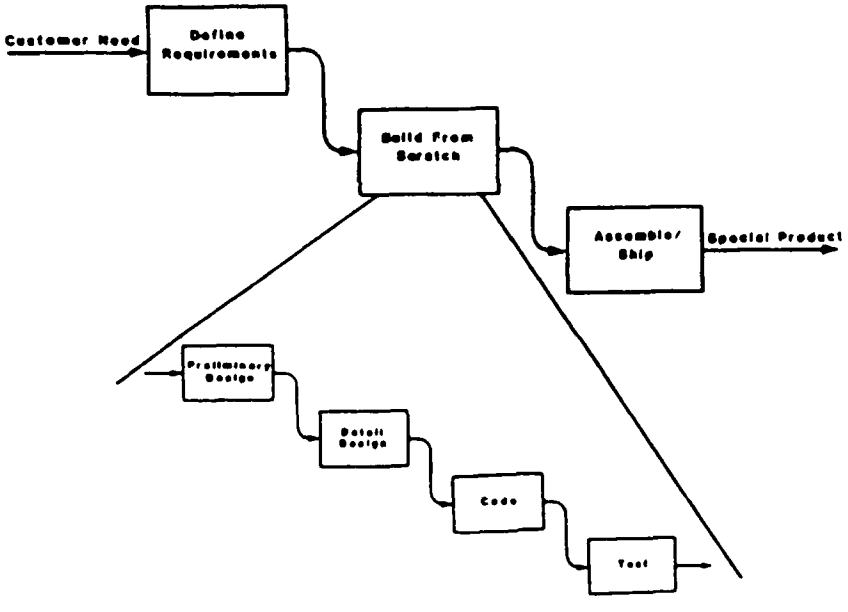


Figure 4

The job shop management methodology is based on a value system that glorifies the job because each job creates a product (from scratch). All management processes reflect this job management philosophy. DP job shops plan by job, e.g., the application portfolio. They organize and staff by job. They market by job. They get funding by job. They measure performance by job and reward job performance. They introduce technology by job. And so on.

The productivity levels that can be achieved in this environment are limited by how fast the organization can build from scratch. This is accomplished formally by automating the basic job management methodology, and informally by people within the organization circumventing the build from scratch job management concept on their own.

Barry Silverman in his IEEE Computer magazine article, "Software Cost and Productivity Improvements, An Analogical View," proposes that this engineering life cycle model, the model around which traditional DP management methodologies are based, "applies to to highly original, one-of-a-kind systems or subsystems." He observes, based on his research, that "the principal weakness of this approach is that while the life cycle model literature and discipline provide a structured approach for those situations in which new products are to be created, they rarely acknowledge the usefulness of analogous products (standard data and program structures) and never tell the software engineer how to incorporate them. For example, though incremental development is referenced, its use is generally described in connection with new products only. Even the literature on software reuse is concerned primarily with new software and how to get it to work on a computer other than the one it is designed for." [SILV 85]

It is difficult to convince DP people that the traditional life cycle model is not cosmic, i.e., the only one. In a sense, it is cosmic in that it is the only way to build stand-alone systems from scratch. In the past systems were stand-alone and built from scratch but in the future they cannot be. The model must change. In fact, according to Silverman, the changing of the model toward an asset-based model, is "a fact of life," based on the economics of production.

The asset-based life cycle model focuses on the development, construction, management, and utilization of assets. It defines two kinds of "projects": 1) asset projects and 2) product projects. Construction methodologies for these two types of projects are different because assets are intended for reuse within product projects, whereas products, produced by product projects, are intended for use by "customers." It requires that assets be created on purpose.

The "reusableness" of an asset is a function of the methodology used to build it. Whether or not the asset gets reused is a function of the product development methodology. It is a fact of life that "assets" bootstrapped on product projects rarely, if ever, get reused. A new, asset-based engineering life cycle model must be evolved as the foundation for both the standard production and customizing shop management methodology. Of course, the new engineering life cycle model is not really new at all. It is very familiar to any business that is run with an asset management style. The basic model is depicted in Figure 5.

Asset-Based Engineering Life Cycle

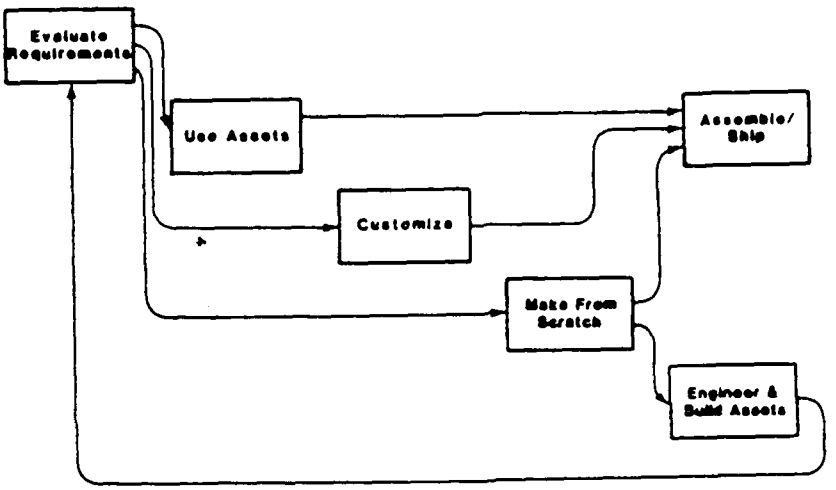


Figure 5

The asset-based life cycle model assumes that the maximum productivity is obtained by servicing requirements that reuse - without modification - assets that have been pre-engineered and pre-produced.

Of course, no practical model could be based entirely on the assumption that all requirements can be serviced by pre-engineered solutions. Therefore, the asset-based model assumes that the second-best choice is to modify pre-engineered solutions. This is called customization.

The model goes on to assume that if reuse is not practical and if customization is not practical, then you should revert to the old model, i.e., build from scratch.

Of course, there would be no concept of reuse or of customization if there were no assets. Thus, assets must be established in a separate life cycle step, which hinges on the build from scratch step. They then must be fed back into the requirements phases so that they can be reused and customized.

The key strengths of the asset-based model are many:

1. It analyzes all requirements as they come in and groups them according to the producibility logic identified above.

2. It forces as many requirements through the reuse and customizing loops as possible - often attempting to restate requirements so that they are more consistent with higher level producibility categories.

3. It schedules and monitors individual sub-projects based upon their producibility category and their interdependence.

4. It systematically analyzes both the customizing and build from scratch loops, looking for new asset opportunities.

5. It contains a separately managed asset engineering, production, and maintenance loop, which not only regularly creates new assets, but it feeds them back to the requirements definition activity. (This activity is based on forecasted ROI that is dependent on the existence of the reuse and customization loops.)

6. It standardizes the language of assets and the language of requirements.

The DP management methodology will not assume the form of an asset-based life cycle model by accident. Obviously, many internal IRM process changes have to take place over time, each one moving the overall management methodology closer and closer to the asset-based model. This transition is condoned and encouraged by a friendly value system. It is enabled by technologies.

Most of today's computer technologies are supportive of the job-based life cycle model. It makes sense. The old management methodologies established the requirements for these technologies. New asset management technologies do not really exist in turnkey form. Pieces of them exist - pieces such as DBMSs, data dictionaries, fourth generation languages, and so on; but these technologies still have not been assembled and refined to support a complete asset-based life cycle model.

THE NEW LIFE CYCLE AND IRM

As you no doubt have anticipated, the new life cycle model dramatically affects how enterprise information resources are managed. The impact is not trivial at all. Indeed, it is changing everything - though not all at once. It affects IRM:

1. Planning
2. Organizing/Staffing
3. Finance/Performance Measurement
4. Standardization
5. Production
6. Order Entry/Master Schedule

The most significant impact on planning, stemming from the asset-based life cycle model, is a balanced market/asset planning philosophy. Such a philosophy characterizes product requirements from a market perspective, and then drives out asset structures that will facilitate production. To arrive at such a plan, DP managers analyze markets; commit to specific, unitized concepts of their end-products; normalize these product concepts to define asset structures; and develop product and asset projects.

Organizing and staffing philosophies fall into line by specifically adding marketing/order entry/master schedule departments and skills, and by adding and supporting asset engineering departments and skills.

Financing strategies and performance measurement strategies are coming to accommodate the separate but interdependent nature of products and assets. Since it has been difficult to finance asset development on product projects, new "capital investment" strategies are to be developed; and since assets are rationalized with ROI logic, performance monitoring is changing to provide the required visibility.

Standardization in an anathema to the traditional life cycle model value system - which is why it has so little effect on current IRM processes. However, standardization of both data and computer technology is becoming a dominant theme in IRM. Standardization will occur at both the "architecture" level and at the "rule" level. Standards are not just nice, they are law, and the systems to control them are well thought out and just.

The production processes in IRM that translate requirements into results are coming into line with the new life cycle model, too. These processes define asset structures and leverage them to increase production. The procedures for defining assets are precise and consistent, and the product development procedures will integrate with them.

And finally, the IRM activities that accept orders and control production schedules are being brought into line with the asset-based methodology. These activities accept large bundles of requirements (from the plans), as well as small bundles of ad hoc requirements, and regroup them to be consistent with the four loops of the asset-based production process: 1) reuse, 2) customize, 3) build from scratch, and 4) build assets. The language of order entry is consistent with the language of assets - to ensure maximum correlation between new requirements and available solutions (assets). The master schedule activities balance and sequence asset and product projects based on their interdependencies.

Obviously, a great many more changes will occur to IRM as the future arrives. The brief glimpse I have provided here will threaten many IRM managers. But, it will intrigue a few of the more enterprising among them. The magnitude of the changes required to imbed the asset-based life cycle model in the information enterprise management methodology is staggering. But, the challenge is no more nor less than that faced by an enterpreneur who starts up a job shop and gets deluged with new business. Success is his greatest risk. It is also his greatest opportunity. If he cannot make the transition to the asset-based life cycle model, he will ultimately go under. So will the IRM manager.

References

[DEAR 72] Dearden, J. "MIS is a Mirage" Harvard Business Review, 1972.

[MCFA 83] McFarlane, W.F. & McKenney, J.L. Corporate Information Systems Management, The Issues Facing Senior Executives. (City, Publisher), 1983.

[SILV 85] Silverman, B. "Software Cost and Productivity Improvements, An Analogical View" IEEE Computer, May 1985.

[APPL 81] Appleton, D.S. "Four Steps to Productivity" Production Engineering, May 1981.

A MODELING SUPPORT LABORATORY FOR LARGE SCALE, DISTRIBUTED, HETEROGENEOUS INFORMATION SYSTEMS DESIGN

Charles R. Standridge

Pritsker & Associates, Inc.

Contributed by Charles R. Standridge

1.0 INTRODUCTION

A large scale distributed, heterogeneous information system is a complex computer-based system having both hardware and software components. Few such systems exist. Thus, little experience has been obtained in designing and implementing these systems. This suggests that computer based aids to assist designers of distributed, heterogeneous information systems would be valuable.

One area of crucial interest in the design of a distributed, heterogeneous information system is performance. Acceptable performance means the system can provide an acceptable level of throughput or response time for an acceptable level of expenditure for software, hardware and their integration. Assessing performance before implementation or without prototyping is important but difficult.

Simulation is an analysis technique which has been shown effective in supporting the design of new, non-existent systems. Typically a formal description of a system called a model is built using the modeling constructs of a particular simulation language. This model is analyzed by a computer program, called a simulator, which manipulates over time a set of variables whose values define the state of the system. The behavior of the system is assessed by observing the state variable values over time.

Recent advances have produced simulation support systems. Typically, a simulation support system consists of graphical builders and forms systems for model entry and update, collection of data during simulation runs, graphics and

report generators to display simulation results, animators to show the structure, dynamics, and control logic of the simulated system, and a database manager to control all project related information.

A modeling support laboratory integrates a simulation language, simulation support system and a host computer. Normally an engineering workstation, such as those provided by Apollo, SUN, or DEC, provides sophisticated graphics and user interface tools to support the hosting of the modeling support laboratory. In addition, constructs are typically added to the modeling support laboratory to support the analysis of the specific problem class of interest.

Possibilities for the development of a modeling support laboratory for designing large scale, distributed, hetrogeneous information systems are discussed. Current simulation language and simulation support system capabilities are presented. Applications of these capabilities in analyzing computer and communication systems are discussed. Potential extensions to existing capabilities are reviewed. The integration of a simulation language, simulation support system, and extensions with application to the design of distributed, heterogeneous information systems will be described.

2.0 EXISTING SIMULATION SOFTWARE

Many simulation languages [2,3,8,9,10,11,16] are in routine use. SLAM II [11] is discussed as an example of a simulation language with an integrated architecture including the process, discrete event, and continuous modeling world views. Several simulation support systems [16] are available. TESS [14,15] is presented as an example system where all software is integrated using a database management system.

2.1 SLAM II OVERVIEW

The discussion in this section is taken from [11] with modifications.

In SLAM II, the alternate modeling world views are combined to provide a unified modeling framework. A discrete change system can be modeled within an event orientation, process orientation, or both. Continuous change systems can be modeled using either differential or difference equations. Combined discrete-continuous change systems can be modeled by combining the event and/or process orientation with the continuous orientation.

The process orientation of SLAM II employs a network structure which consists of specialized symbols called nodes and branches. These symbols model elements in a process such as queues, servers, and decision points. The modeling

task consists of combining these symbols into a network model which pictorially represents the system of interest. In short, a network is a pictorial representation of a process. The entities in the system (such as people and items) flow through the network model.

In the event orientation of SLAM II, the modeler defines the events and the potential changes to the system when an event occurs. The mathematical-logical relationships prescribing the changes associated with each event type are coded by the modeler as FORTRAN subroutines. A set of standard subprograms is provided by SLAM II for use by the modeler to perform common discrete event functions such as event scheduling, file manipulation, statistics collection, and random sample generation. The executive control program of SLAM II controls the simulation by advancing time and initiating calls to the appropriate event subroutines at the proper points in simulated time. Hence, the modeler is completely relieved of the task of sequencing events to occur chronologically.

A continuous model is coded in SLAM II by specifying the differential or difference equations which describe the dynamic behavior of the state variables. These equations are coded by the modeler in FORTRAN by employing a set of special SLAM II defined storage arrays. The value of the Ith state variable is maintained as variable SS(I) and the derivative of the Ith state variable, when required, is maintained as the variable DD(I). The immediate past values for state

variable I and its derivative are maintained as SSL(I) and DDL(I). respectively. When differential equations are included in the continuous model. they are automatically integrated by SLAM II to calculate the values of the state variables within an accuracy prescribed by the modeler.

As an example of a network model, consider the model of a teller in a bank shown in Figure 1.

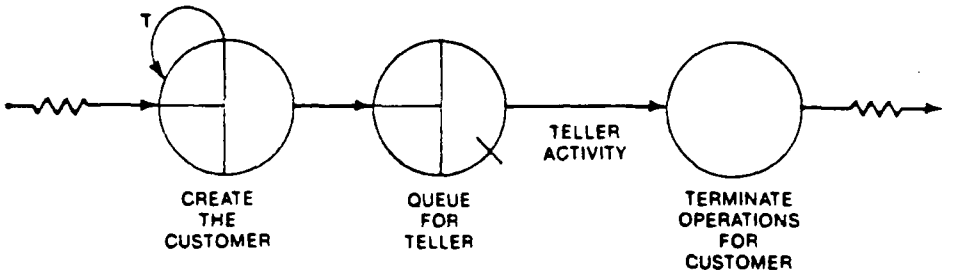


Figure 1. Network of Bank Teller Model

The first node creates the arrivals of customers to the system with a time between arrivals of T time units. T could be specified to be a constant or to be a sampled value. The QUEUE FOR TELLER NODE specifies that the entity is to wait until the teller is idle. The TELLER ACTIVITY models the elapsed time during which the customer is served by the teller. Following completion of service. the customer is terminated from the system. The teller is then available to process any waiting entities at the QUEUE FOR TELLER NODE.

From the above example, we see that the process orientation provides a description of the flow of the entities through a process. Its simplicity is derived from the fact that the event logic associated with the statements is contained within

the symbols of the simulation language. However, since we are normally restricted to a set of standardized symbols provided by the simulation language, modeling flexibility is not as great as with the event orientation.

2.2 TESS OVERVIEW

TESS provides an integrated environment for performing simulation projects by supporting modeling, simulation and analysis, and result presentation as shown in Figure 2. All capabilities revolved around a single database where all project information is stored. The TESS language provides access to all capabilities.

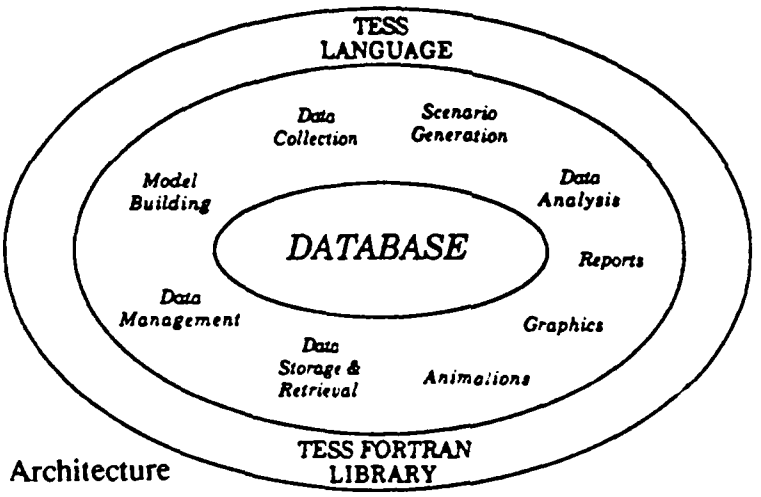


Figure 2. TESS Architecture

Modeling tasks are supported by graphical builders for entering and editing SLAM II networks, descriptive models called facilities and elementary symbols from which facilities are composed called icons.

Simulation and analysis tasks are supported in various ways. A forms system aids in the input of simulation run controls. Simulation results including event traces, observations of performance measure time series, and statistical summaries are automatically collected, organized and stored in the TESS database. Results from multiple scenarios are stored in the same database for future across scenario comparisons. Computation independently of simulation runs of summary statistics, frequency distributions and confidence intervals is provided. Furthermore, data may be unloaded from the database into sequential file form so that complex statistical analyses may be performed using existing statistical packages.

Data selection mechanisms provide for the flexible selection of simulation results to appear on reports and graphs. Scenarios, performance variables, simulated time periods of interest and arbitrary logical conditions in the data can be used to select data for reporting and graphing. Spike plots are used to show time series of non-time persistent variables such as time in the system. Discrete plots show discrete time persistent data such as queue lengths. Continuous plots show variable values which can change between events. Bar charts, histograms, and pie charts show frequency distributions. Range charts show averages, minima and maxima. Reports show data values and statistics.

Traces of simulations and of actual system behavior are stored in the TESS database. Alternatively, simulation traces can be sent directly to the TESS animator so that animations may be displayed concurrently with simulations. Icons and

facilities are built with graphical builders and stored in the TESS database. A collection of animator commands, called a rule, relates the event occurrences of a trace to the actions of an animation. A rule is built using the forms system and is stored in the TESS database. The animator uses a facility, trace and rule to generate an animation. The pause processor provides for user interaction with the animation to dynamically query the animation status or make ad hoc changes in the simulation.

The TESS language consists of a set of statements for specifying model building, statistical analysis, reports and graphs. The TESS input system assists the user in constructing statements through prompting for required information, display of allowable options, and help facilities. Parameterized sets of statements called macros can be constructed to provide application specific statements.

3.0 COMPUTER AND COMMNICATION SYSTEM APPLICATIONS OF SIMULATION

Many articles concerning computer and communications systems modeling and simulation are found in the literature. Due to space limitations, three involving Pritsker & Associates, Inc. will be discussed.

Standridge and Phillips [12] modeled the communications from each member of a package of experiments designed for use aboard the space shuttle, to a single microprocessor controlling the package, and finally to a receiving station on earth. Algorithms for assuring that each experiment received an adequate and proportional share of communications resources (microprocessor memory and communications ports) were tested. Constant rates of transmission and large burst transmissions for image processing were evaluated. The adequacy of the microprocessor memory size and number of ports were assessed.

Standridge and Duket [13] describe a model assessing a communication network including satellite communications. The network transfers messages between message centers directly and through a distribution facility which performed the administrative functions of the system. Each message center has one transmission subsystem for accepting, coding and transmission of originating messages as well as one processing subsystem for storing, decoding, processing and rerouting incoming messages. The model was used extensively by communication system personnel to evaluate various system design alternatives.

Musselman and Hannan [4] describe a model of a distributed data processing network used by the Bank of America to support state-wide on-line banking operations. This complex interactive network is controlled by two computing

centers. Information is transmitted by means of communication channels between the computing centers and branch banks located throughout California. Each branch has a programmable control unit which regulates teller access to these channels. The model was used to evaluate response time versus utilization trade offs in the system.

4.0 POTENTIAL SIMULATION CAPABILITY ENHANCEMENTS

Simulation languages and support systems need to be easier to use for all users in order to broaden the acceptance and use of simulation techniques. Clearly, a software tool is needed which provides computer assistance in using simulation languages and simulation systems. The following characteristics should be included in this tool:

1. Computer assistance in model conception - the ability to relate system entities to modeling language constructs in model building as well as the use of user defined icons in constructing models.

2. Computer assistance in model entry - graphical and/or forms oriented builders for entering and editing models, checking the validity of parameters and the completeness of models and their acceptability for simulation.

3. Computer assistance in model simulation - derivation of data collection specifications and other simulation experiment parameters from study goals as well as automated responses to events and conditions detected during the simulation.

4. Computer assistance in animation - the use of model icons in deriving facility diagrams and the derivation of animator commands from system information and model icons as well as identification of time periods, conditions and events of critical interest based on the goals of the study.

5. Computer assistance in result presentation and analysis - derivation of performance measures from project goals from which appropriate statistical analyses, graphs and reports can be determined.

6. Computer assistance in conclusion drawing - derivation of insightful statements about system behavior based on project goals and simulation results including the suggestion of alternative scenarios.

7. An open set of modeling constructs - allow for user constructed modeling constructs tailored to the class of systems the user studies.

8. A hierarchical system view - allow for the decomposition of entities into a components and the aggregation of components into entities in both modeling and all simulation results processing.

9. User-in-the-loop architecture - the software must confine itself to making suggestions to the user or allow the user to override any automatic actions, that is the software must not prevent the user form doing what the user wants.

Existing technology helps in building tools to meet these requirement. Technology which could be used to address points 1-9 above will be discussed.

4.1 MODULAR MODELING

Modular modeling has long been advocated in the literature [1,5,6,7,17,18] as well as having been implemented in artificial intelligence based simulators and continuous simulators. The principles of modular modeling are simple.

1. A set of model modules augments the modeling primitives of a simulation language to provide the available modeling constructs.

2. A model module is built from other model modules and/or the modeling primitives.

3. The modeling primitives are the "given" of the system.

4. Each module may have parameters specified when the module is used.

The modular modeling processor (MMP) is software which is added to the simulation language. Modules including references to modules are built. The MMP composes models which can be analyzed by the simulator from the modules.

The simulation support system is extended to assist the modular modeling task. Modules are stored in the simulation support system database. Two new graphical builders are added. The module icon builder provides for the drawing of icons with user defined parameters which correspond to modules. Such icons can be used in the graphical construction of models using the new model builder. A graphical model may be parameterized and represented by an icon for use as a module in another model.

The composition of models from component modules provides an open-ended, flexible and usable approach to model construction. New model modules can be added to the modelbase and existing model modules modified at any time by anyone familiar with the standard simulation language in which the model modules are implemented. Models are constructed by selecting the appropriate model modules and specifying their parameters. Thus, only a knowledge of what modules are available is necessary for model building, supporting model builders who know the capabilities of the modules, but aren't expert in the underlying simulation language.

As an example, consider an example module which sends a packet of data to a receiving site.

```
   MODULE TRANSMIT (SIZE, QMEMORY, MEMORY)
*  SIZE IS THE PACKET SIZE IN BYTES
*  QMEMORY IS THE QUEUE ID OF THE MEMORY AT THE RECEIVING SITE
*  MEMORY IS THE MEMORY AT THE RECEIVING SITE
   GLOBAL RATE                                  ACCESS TRANSMISSION SPEED
   AWAIT (QMEMORY), MEMORY/SIZE;                WAIT FOR MEMORY AT
                                                RECEIVING SITE
   ACT, SIZE/RATE;                              TRANSMIT PACKET
   GOON;                                        END OF TRANSMISSION
   ENDMODULE;
```

A standard simulation language, SLAM II in this case, statements (AWAIT, ACT, GOON) can be parameterized and mixed with statements which direct the composition of models from modules (GLOBAL).

4.2 ARTIFICIAL INTELLIGENCE

Artificial intelligence/expert systems techniques show promise in providing a tool kit for building software support for tasks now delegated to the user. To date, this promise has not been fulfilled. Several artificial intelligence technology based companies have entered the simulation arena. Typically, their simulation technology has been lacking and their application of artificial intelligence to the simulation process naive. This has resulted in products which are more prototypes than stable, commercial products.

However, these products have demonstrated important concepts in computer assistance in model conception, model entry, model simulation and conclusion drawing as well as allowing user defined modeling constructs. A user defined modeling construct may correspond exactly to a system entity minimizing model conception tasks. For example, a modeling construct may exist for accessing a record within a file. Minimal conditions for model completeness (the model can be simulated) are in the knowledge/rule base and can be used to check all models. A list of pending activities for model completeness is maintained. The simulation of the model is goal-directed. Based on the user-specified goal, significant simulation results are identified and collection of these results specified automatically.

On the other hand, a lack of sophistication in existing AI/Simulation software for analysis, animation, graphing and reporting gives few suggestions for computer assistance in these areas. In addition, these products have been criticized for requiring a high-set up cost before problem solving. Someone must build and maintain a large knowledge/rule base including modeling constructs, acceptable goals, and rules for drawing conclusions from simulation results. These efforts may be more costly than the use of conventional simulation technology and may outweigh the benefits returned.

Potentially, five AI-based enhancements can be made to simulation support system. Model builders are enhanced with computer assistance in model conception and entry. Users are allowed to specify a system component and receive

information about what model modules and primitives would best model that component. Models are checked for completeness and consistency of parameter values. Information required from the user but not yet specified will be tracked by the software.

The experimental control generator automatically produces as much of the experimental control, including the data collection specification, as possible. The control is stored in the database where it can be revised or even replaced using existing capabilities. The experimental control generator is guided by the contents of the knowledge base.

The animation generator, guided by the contents of the knowledge base, generates a diagram on which to show the animation, from the model of commands of the system and a set of commands for animating the simulation. Both the diagram and the commands are stored in the database for further processing using existing capabilities. Methods of generation need to be investigated.

The result examiner/conclusion drawer assists the user in applying the simulation results. This component must converse with the user in terms of the system being modeled and translate user specifications into queries resulting in graphs, reports, statistical analyses, suggestions for new scenarios and/or qualitative conclusions about system behavior. Strategies and techniques required for this component are not well known and much further study is required.

The knowledge/rule base manager controls the directive information needed by the control generator, the animation generator, and the result examiner/conclusion drawer. Information such as the goals of the study and rules about making inferences from results must be user supplied. Other required information must be identified. User interfaces for entering and maintaining this information must be provided.

5.0 A MODELING SUPPORT LABORATORY

Existing technology such as simulation languages, simulation support systems and workstation computers enhanced by modular modeling and AI-based advances, can be integrated into a modeling support laboratory for the designers of large-scale, distributed heterogeneous information systems.

Modeling of these systems can be supported by both a general purpose simulation language, such as SLAM II. and a library of model modules. Each model module would represent a component of the information system such as the TRANSMIT component modeled in the previous section. Thus, models could be constructed in terms of information system components facilitating the speed and ease of modeling. Modules would be developed in the general purpose simulation

language facilitating their initial construction and evolution over time. Assistance in the entry of such models can be provided within the simulation support system. A graphical builder in which icons are used to represent modules can be used to enter models. AI-based techniques can be used to make knowledge about the relationship between modules and the system components they represent available in the builder. This allows the modeler in the builder to ask questions such as "How could I model the transmission of data?".

The simulation support system helps in the analysis and presentation of simulation results as well. Results from all simulation runs are preserved in the support system database so that across scenario analysis and other comparisons between scenarios can be made. Reports and graphs can be used to examine the simulation results. Animations present the dynamics of the simulation. AI-based techniques can be used to assist in the construction of these presentations. For example, the modeler could request to see the number of transactions against a particular database overtime. The support system could respond by displaying a line graph of the number of transactions over time and a pie chart of the distribution of the number of transactions by time of day. Assistance in conclusion drawing could be provided as well. For example, the query "Identify the computer which is the bottleneck in query processing" could result in a search of the database for the computer with longest average response time to queries and the printing of the result.

BIBLIOGRAPHY

1. Ketcham, Michael G., John W. Fowler, Don T. Phillips, "New Directions for the Design of Advanced Simulation Environments", Proceedings of the 1984 Winter Simulation Conference, December 1984, pp. 563-568.
2. Kettenis, D. L. "Combined Simulation with SIMULA '67" in Simulation of Systems, North-Holland, 1979.
3. Law, Averill M. and Christopher S. Larmey, An Introduction to Simulation Using SIMSCRIPT II.5, CACI, 1984.
4. Musselman, Kenneth and Robert J. Hannan, "A Network Simulation Model of a Computer Communications System". Proceeding IIE Spring Conference, May 1984.
5. Nance, Richard E., Ahmed L. Mezaache, C. Michael Overstreet, "Simulation Model Management: Resolving the Technological Gaps", Proceedings of the 1984 Winter Simulation Conference, December 1981, pp. 173-179.
6. Oren, Tuncer and Bernard Zeigler, "Concepts for Advanced Simulation Methodologies", Simulation, Volume 32, No. 3, March 1979, pp. 69-82.
7. Oren, Tuncer, "Computer-Aided Modeling Systems", Progress in Modeling and Simulation, PP. 189-203, ACADEMIC PRESS, 1982.
8. Pegden, C.Dennis, "Introduction to SIMAN", Proceedings of the 1983 Winter Simulation Conference, December 1983.
9. Pritsker, A. Alan B., Modeling and Analysis User Q-GERT Networks, Halsted Press, 1977.
10. Pritsker, A. Alan B., The GASP IV Simulation Language, John Wiley & Sons, 1974.

11. Pritsker, A. Alan B., Introduction to Simulation and SLAM II, Third Edition, Halsted Press, New York and Systems Publishing Corporation, West Lafayette, Indiana, 1986.

12. Standridge, Charles R. and James R. Phillips, "Using SLAM and SDL to Assess Space Shuttle Experiments", Simulation, July 1983.

13. Standridge, Charles R. and Steven D. Duket, "Applications of Database Management and Graphics in Simulation", Modeling, December 1984.

14. Standridge, Charles R., "Performing Simulation Projects with the Extended Simulation System", Simulation, December 1985.

15. Standridge, Charles R. and A. Alan B. Pritsker TESS: The Extended Simulation Support System, Halsted Press, 1987.

16. Wilson, James R., James O. Henriksen and Stephen D. Roberts, Proceedings of the 1986 Winter Simulation Conference, December 1986.

17. Zeigler, Bernard P., "Concepts and Software for Advanced Simulation Methodologies", Simulation with Discrete Models a State-of-the-Art View, December 1980, pp. 25-33.

18. Zeigler, Bernard P., Multifaceted Modeling and Discrete Event Simulation, Academic Press, 1984.

END

DATE

FILMED

9-88

DTIC