1-1-1994

Systems Analysis

Richard O. Mason
Southern Methodist University

Sue A. Conger
Southern Methodist University

In ENCYCLOPEDIA OF OPERATIONS RESEARCH AND MANAGEMENT SCIENCE, Saul Gass and Carl Harris, Editors, Newell, MA: Kuhler Academic Publishers, forthcoming, 1994.

SYSTEMS ANALYSIS

Working Paper 94-0402*

by

Richard O. Mason
Sue A. Conger

Richard O. Mason
Sue A. Conger
Edwin L. Cox School of Business
Southern Methodist University
Dallas, Texas 75275

* This paper represents a draft of work in progress by the authors and is being sent to you for information and review. Responsibility for the contents rests solely with the authors and may not be reproduced or distributed without their written consent. Please address all correspondence to Richard O. Mason.

Systems Analysis

Richard O. Mason
Sue A. Conger
Southern Methodist University

History and Philosophy: Systems analysis is a broad term applied to the study of real world processes. It requires breaking problems down into their component parts and formulating a conceptual definition of the situation. The purpose is to develop "an overall understanding of optimal solutions to executive type problems" (Churchman, et al., 1957, p. 7). The resulting conceptual definition often is subsequently translated into a computer-based system or mathematical model. Systems analysis has been applied to understand complex, dynamic systems -- both physical and social -- such as business processes, governments, economics systems, weapons systems, mechanical and manufacturing systems, and computer software. The science is based on several key theories: systems, cybernetics, and modeling.

A system is a set of related elements organized to achieve a purpose. The elements form "an interconnected complex of functionally related components." (Churchman, et al., 1957, p. 7) Each element has *inputs, processes, and outputs*. At the most detailed and fundamental level of analysis, the elements are generally treated as "black boxes". At a high level of abstraction, what goes into and out of each black box element is described but the activities within the element are not described. Each black box is analyzed in turn to define the *transformation* process on its inputs that generate its outputs. The concepts of *flow, relationship, message,* and *connection* are used to portray the *structure* of a system which has been analyzed. These terms describe the interrelationships among its elements. The transformation processes are described in terms including *transaction, process,* and *problem.*

A cybernetic connection integrates a feedback loop into a system, and provides communication about the system's outputs which are used, in turn, to make adjustments in either the inputs or the process necessary

to achieve the system's purpose. This is called *control* (Weiner, 1948). (See "Cybernetics in Systems Management" in this volume.)

A mathematical model of a system is a "collection of mathematical relationships which characterize the feasible programs" for improving a system (Dantzig, 1963). Building a mathematical model provides insight into a system and its properties. The model can also be manipulated to derive conclusions about the system.

Mathematical models and other OR/MS techniques may be applied to the conceptual definition of a system and used to determine the best possible solution -- the optimum decision, policy, or design for the system -- for the problem the system represents. Churchman poses five necessary conditions for completing a systems analysis:

1. The total system objectives and, more specifically, the performance measures for the whole system

2. The system's environment: the fixed constraints (which are 'outside' the system)

3. The resources of the system (which are capabilities found 'inside' the system and, therefore, can be reallocated)

4. The elements of the system, their activities, functions, goals, and measures of performance

5. The management of the system. That is, the process of allocating resources to the system's elements. (Churchman, 1968).

Most systems contain recognizable sub-systems, sub-sub-systems, and so on, organized in a hierarchy. Their arrangement often involves a "Chinese box" form of nesting that permits them to be defined by recursion. A solution which is best for the system as a whole is called an 'optimum'; whereas, one which is best relative to the functioning of one or more elements is called a 'suboptimum.' One of the challenges of systems analysis is to improve the performance of a sub-system in terms of its own goals and purposes -- sub-optimization -- without detriment to the total system or, worse, defeating the system's overall purpose.

Systems Analysis and Computer Systems: Systems analysis is the first stage in presenting any large task to a computer (the other principle stages being design and implementation). It is performed by a systems analyst and consists of analyzing the whole task in its setting and deciding how to arrange it for processing by a computer. It includes an estimation of how much work is involved and hence how powerful a computer will be required. The problem is divided into a number of relatively independent parts which are specified, together with their interconnections, in sufficient detail for a programmer to take over.

Computer applications are developed through a series of translations. The first translation, as noted above, is from a real world situation to a conceptual definition of the situation. This conceptual model is then translated via a design activity to an implementation model that is still readable by humans and describes the conceptual model in a language related to the target computer environment. The implementation model is then translated into the specific coded language(s) of the target (machine) environment. These three translations define phases of activity that comprise an application's *development life cycle*. The translations relate to the thinking processes involved and are called *analysis, design,* and *implementation*, respectively. Implementation can be divided into sub-phases for programming, testing, and cut-over.

Software development methodologies are used to guide the development processes through the life cycle. (Technically, *methodology* is the study of tools and techniques, and *methods* are tools and techniques. The common term for system development methods used as a package of tools is methodologies and it is used here.) Five different approaches are used currently: mathematical, process, data, object, and artificial intelligence. All use top-down strategies for problem-solving and progressively decompose a large problem into smaller, solvable problems for independent solution (Laszlo, 1972). The five approaches can be further divided into three classes -- mathematical, transaction and semantic -- depending on the type of problem they attempt to solve (see Figure 1). Mathematical methodologies solve selection and alternative analysis

problems. Process, data and object methodologies solve transaction processing problems. Semantic methodologies solve reasoning problems.

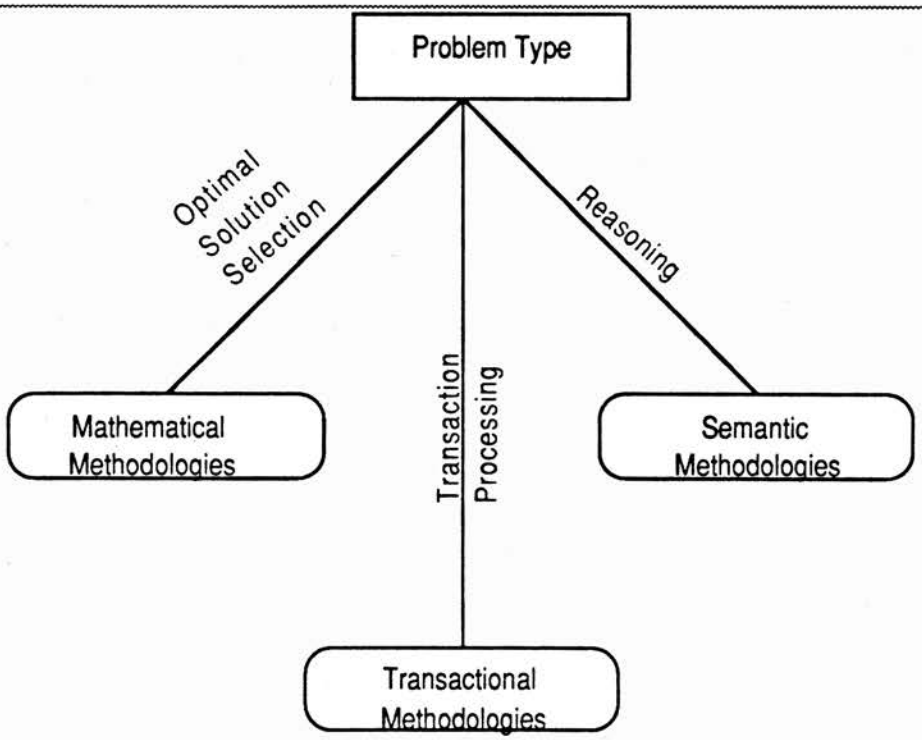


Figure 1. Methodology Classes

Mathematical Methodologies: Mathematical methodologies employ mathematical models of a system and focus on the logical relationships within the system. They are often formulated by interdisciplinary teams, who adapt scientific theories and methods to practical problems (Ackoff & Rivett, 1963). Operations research (OR), management science (MS), and cybernetic methods are applied. Classes of problems to which mathematical methods apply include inventory, allocation, sequencing, queuing, routing, replacement, competition, and search (Ackoff & Rivett, 1963, p. 34). The problems solved by OR techniques all deal with selection from many alternatives and sensitivity analysis to develop alternative, contingent courses of action.

Mathematical, cybernetic systems seek optimal solutions based on unambiguous, but possibly incomplete, information (Churchman, 1957).

The inputs to mathematical applications define the alternatives and resources available from which an optimal selection must be made. The tools and techniques used to develop mathematical models include linear, network, dynamic, and stochastic programming methods (Wagner, 1975). The results of these applications are usually presented in the form of suggested machine schedules, resource allocations, and so on.

Transactional Methodologies: Transactional methods focus on the flows of information between the elements of a system. Three different methodology classes have evolved to develop transactional, information retrieval, and data analysis types of applications: process, data, and object. No single methodology currently supports all three application types well. Further, as the demand for client/server systems and distributed systems evolve, improved methodologies will be required to support their development.

Process Methodologies. Process methods developed during the 1950s and 1960s to mirror Von Neumann computer architecture which separates inputs and outputs from processes. Since computing was the difficult issue at the time, processing was the focus of process methods. In particular, the types of problems automated included accounting procedures, order entry, inventory processing and so on. These applications all deal with transactions that support the basic white collar operations of the organization.

The development techniques focus on data flowing between processes (DeMarco, 1979; Yourdon, 1979). The sample data flow diagram in Figure 2 shows the processes as circles connected via directed lines (i.e., data flows) to external entities or data stores. External entities are depicted on the diagram as squares and represent people, organizations, or other computer systems from which and to which information flows. Data stores, on the diagram as open-ended rectangles, indicate files of information that persist over time. The lines connecting the other icons indicate temporary data flowing through the system, hence the term, data flow diagram.

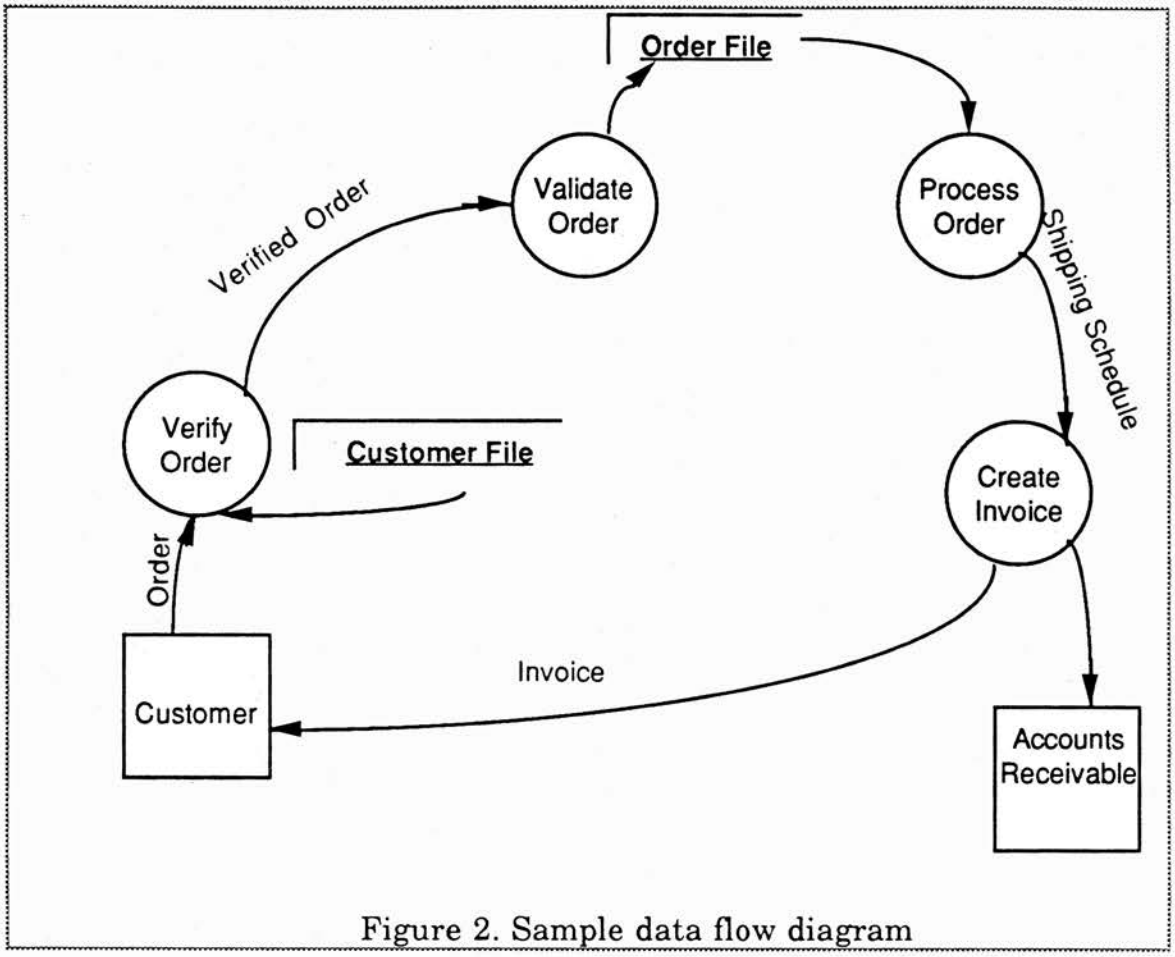


Figure 2. Sample data flow diagram

Process methodologies and methods have undergone several iterations of refinement to support real-time systems development (Ward & Mellor, 1985, 1986). The lack of integration of data throughout analysis and design has led to an abandonment of process methods, *per se*, in favor of techniques which provide such integration.

Data Methodologies. Data methodologies developed as database technologies that came to the market in the 1960s and 1970s were found to require specific attention to data design. Data methods are based on theories of semantic modeling (Chen, 1981), relational database design (Codd, 1972), and data normalization (Kent, 1983). These theories are significant in business because they result in mathematically, provably correct processing of data, a key in mission critical applications. They are also significant because they signal the application of mathematical

foundations for transaction processing which, before this, relied on analyst and programmer ingenuity and accuracy.

The essence of relational data design is that information should look to the user as if it were composed of rows and columns, similar to a spreadsheet (see Figure 3). The physical implementation should be transparent, and entity and referential integrity must be maintained. *Entity integrity* refers to primary keys as a unique identifier of a relation and states that no component of a key may accept null values (Date, 1990). *Referential integrity* guarantees that no relation contains unmatched foreign key values. A foreign key is a primary key in one relation that appears as an attribute, or foreign key, in another relation (Date, 1990).

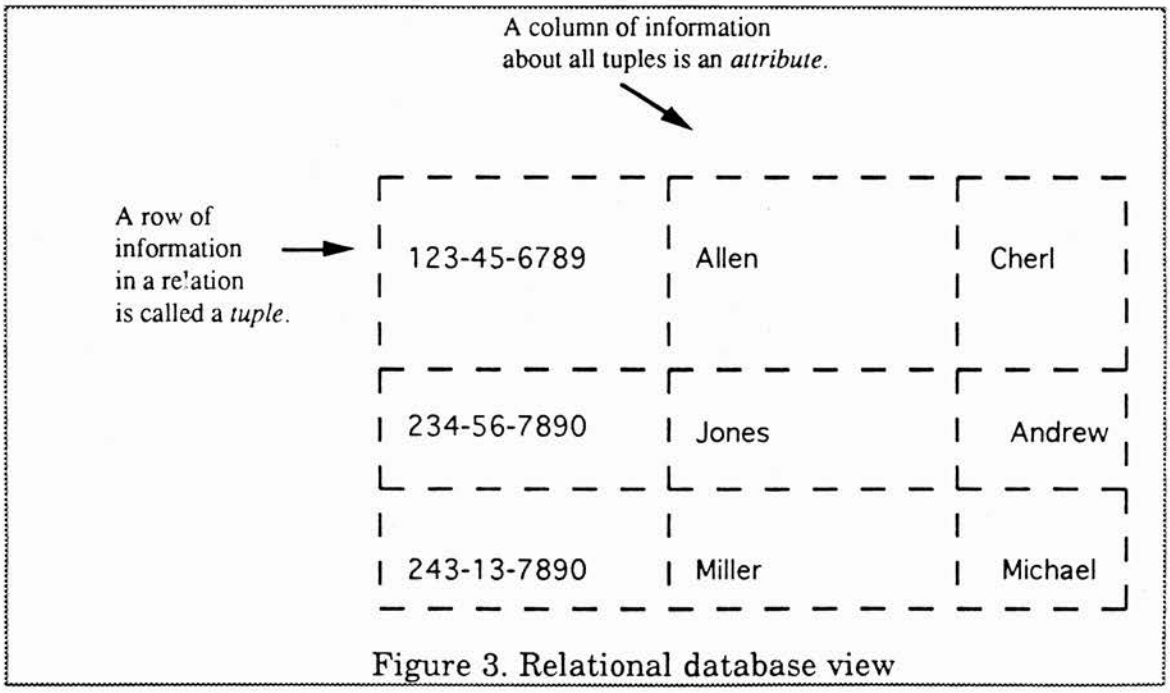


Figure 3. Relational database view

Early data methods focused only on data with the assumption that all primitive processing - create, retrieve, update, and delete - followed logically from the correct definition of data (Warnier, 1981). Demands to capture the complexities of the world led to significant extensions of process data flow analysis and the development of data modeling with integration of data and process throughout the methodology. This is called information engineering (IE) (Martin & Finkelstein, 1981). Data models in the form of entity-relationship diagrams, process models similar in form to data flow

diagrams, and integration models that links data and process are all found at each stage of information engineering (see Figure 4).

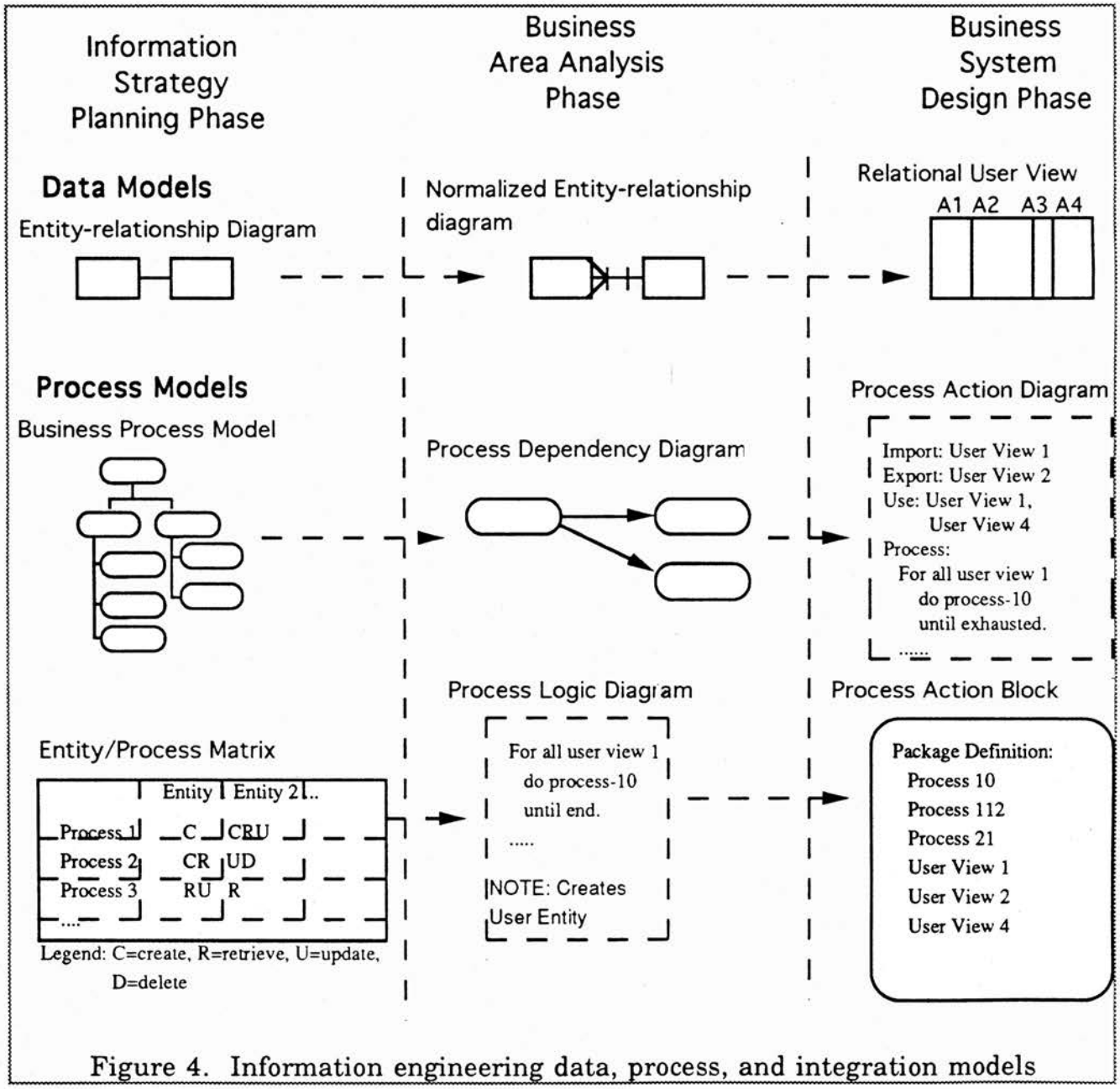


Figure 4. Information engineering data, process, and integration models

Information engineered applications are assumed to use traditional processing languages with integration of database technologies. Computer aided software engineering tools that support the development of IE applications also generate code that imbeds relational database code within it. Data methodologies assume on-line applications but can be used for

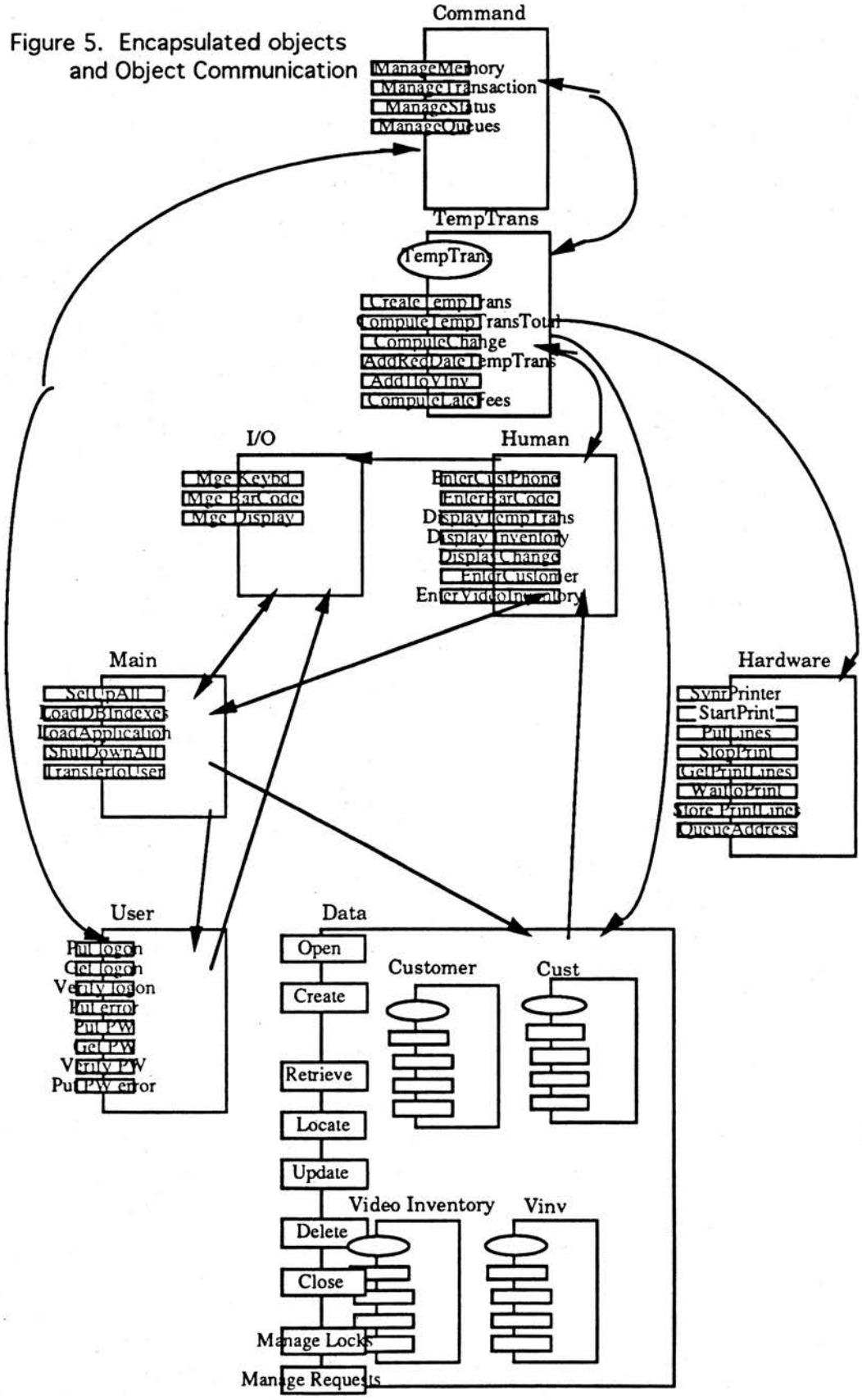
batch processing as well. They are less adapted to real-time applications. Data methodologies are widely used in large, Fortune 500 organizations that rely on databases having millions of relations.

Object Methodologies. Object techniques were originally formalized at Xerox PARC in the 1970s with the development of Smalltalk™ and eventual commercialization of the Apple Lisa™. As on-line and real-time technologies migrated from the aerospace and defense industries to commerce, improved methods were needed to explain how elements in a system interacted with one another. Object-oriented analysis (OOA) involves development of three models: (1) an information model for describing elements in terms of objects and attributes, (2) a state model which describes the behavior of objects and relationships over time, and (3) a process model which specifies the actions in terms of elementary and reusable processes (Schlaer & Mellor, 1992).

The goal of object methods is complete integration of data and processes in *encapsulated* objects (see Figure 5). Objects may be members of *classes* and exhibit *inheritance*, a property such that properties, data and processes of related objects may be reused without redefinition, that is, inherited (Coad & Yourdon, 1990; Coad & Yourdon, 1991). Objects may have multiple inheritance from competing objects throughout a hierarchy (see Figure 6). Objects may also exhibit *polymorphism*, that is, the ability to have the same process, using one public name, take different forms when associated with different objects (Booch, 1987; Booch, 1991). Client/server technology embodies the concepts of object orientation. *Server objects* perform a requested process; whereas, *client objects* request a process from a supplier.

Object orientation is based on the same theories as data methodologies, carrying normalization to the encapsulated (data + process) object units (Kent, 1983; Kim, 1989). The most visible example of object applications is MS Windows which uses windows, icons, menus and pointers in an object-oriented human interface to personal computers. Object-oriented methodologies are currently adopted widely in the embedded systems and software markets (e.g., graphical user interfaces - GUIs such as MS Windows).

Figure 5. Encapsulated objects and Object Communication



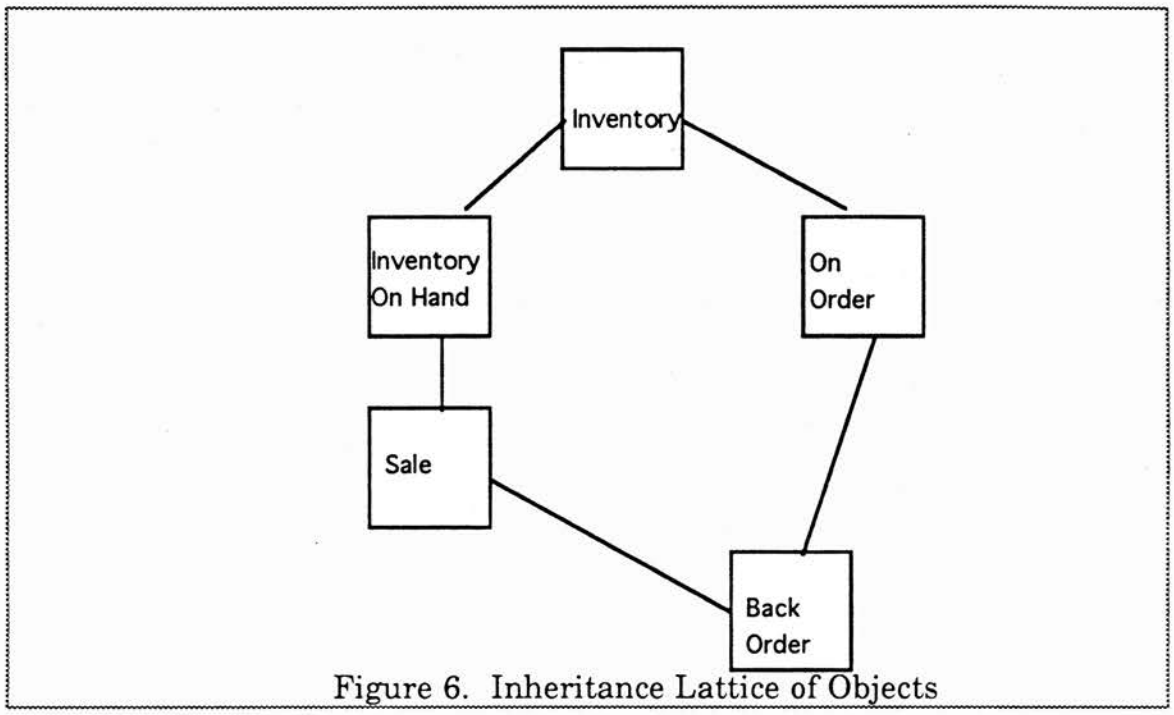


Figure 6. Inheritance Lattice of Objects

Object methods are in the experimental stage today and are running into obstacles that appear intractable in the short-run. The traditional Von Neumann architecture separates data from the programs which process them but, object encapsulation assumes non-persistent data that are discarded upon use. Many applications, however, require data persistence. The problem revolves around the need for persistence without massive duplication of processes that occurs with strict encapsulation. Object-oriented databases have solved some of the design problems by imbedding references to procedures within the database schemas that define the data, making programmed procedures another element in the database. Speed and efficiency of processing for current object databases, plus the added requirements of auditability, reliability, and, eventually distributability have not been solved at this time (Kim & Lochovsky, 1989).

A further problem is that object-oriented methods do not have adequate standardized procedures or common definitions of terms. The fact that they are closely coupled to implementation languages requires that analysis and design methods be based on a desired target language. Some authors suggest C++ object-oriented analysis, others suggest Ada object-oriented analysis, while others suggest their favorites. These issues will

preclude object-orientation from eclipsing other methods of application development for business applications in the near future.

Semantic Methodologies. *Semantic* refers to *meaning*. Semantic methods focus on the role of knowledge and meaning in a system. They imbed meaning in data and in reasoning rules used to process it by drawing on theories of cognitive development as it applies to computer reasoning and learning (Anderson, 1987; Kolodner, 1986). Semantic methodologies, such as Artificial Intelligence (AI), are used to design computer systems which understand languages, learn, reason, solve problems and exhibit other characteristics associated with intelligence in human beings (Barr, 1981). They differ from mathematical and transactional methodologies because they produce a decision, a course of action or an answer to a question based on the application of qualitative knowledge and information. The more qualitative the systems situation being examined, therefore, the more advanced the techniques required. For instance, whether or not a person has toast for breakfast in the morning might depend on the previous night's sleep, one's dinner, recency of exercise, and so on. This simple example illustrates two of the problems AI applications must solve: making their reasoning sufficiently generic and identifying all of the relevant quantitative and qualitative relationships in the system.

A program called DENDRAL was an early result of AI research. It is a chemist's assistant that interprets data from a mass spectrograph and infers the chemical structure of an unknown organic compound. The program is based on an algorithm developed by J. Lederberg in 1964 which generates all possible acyclic graphs given the number of systems elements (the compound's chemical composition) and the number of links (relationships) pertaining to each element (the technical valences). The number of possibilities generated for any given compound is enormous. In order to avoid an exponential search, DENDRAL automates rules to apply heuristics and knowledge gained from practicing chemists to delimit radically the number of alternatives that must be evaluated to determine the compound's molecular structure. DENDRAL introduced the idea of using *rules* to represent expert knowledge, a concept that has prevailed in AI work since (Feigenbaum, 1971). Today, DENDRAL outperforms expert

chemists on this task (Buchanan & Feigenbaum, 1981; Churchman, 1971; Feigenbaum, et al., 1971; Smith, et al., 1973). In another stream of groundbreaking research, Minsky and Papert pioneered the application of parallel processing ideas to reasoning problems (Minsky, 1968; Papert, 1980).

Neural networks, the newest AI technique, takes the human brain as the system under analysis and attempts to model human intellectual activity on a broad scale by mirroring human brain functioning. A *neuron* is the smallest possible processing element and is related to other neurons via *synapses*. Objects called *dendrites* are message transmitters that flow between neurons over synaptic connections (Zahedi, 1993). Single neurons can have thousands of synapses. Inputs via dendrites can either excite (i.e., initiate) or inhibit action of a neuron. The number and frequency of messages to a neuron creates an activity level that can be triggered when some predefined threshold is reached. Each neuron has *axons* via which output signals are transmitted to the dendrite network (Zahedi, 1993). These terms have parallels in the other methodologies but work slightly differently in neural nets. Neural networked problems, however, are different in kind from those solved by the other methodologies. AI and neural problems deal with incomplete information, probabilistic outcomes, and ambiguities in the reasoning and data to be used in the development of a solution.

The techniques that comprise semantic methodologies are not really mature enough to be called methodologies. Rather they are individually applied, taught in an master-apprentice relationship, based on practical experience with a given set of problems.

At this time, different types of reasoning problems require different types of methods and approaches to automating intelligence (Winograd, 1986). The problem types which are addressed by semantic methods include language understanding and translation, sensory understanding (i.e., sight, touch, etc.), memory recall and forgetting, and coordination and control of movement. The most common of these are expert systems which exhibit intelligence in selecting an action by reasoning through numerous, sometimes contradictory, rules. Expert systems have found acceptance in industry and government for applications such as surveillance of nuclear

plant operations, selection of geological drilling sites, diagnosis of medical problems, and so on.

AI techniques and methodologies are in an emergent stage, experiencing continuous refinement and evolution. Like object-oriented methods, AI methods are also closely coupled to the target implementation language. For instance, some languages require data integration with reasoning rules while others require separation of data from reasoning rules. Most languages offer one reasoning approach which determines the nature of the reasoning process as forward, backward, depth-first, breadth-first, custom-defined, or other. The major business promise of AI is to augment existing applications to include reasoning about the processes and data they maintain. Neural nets are promising as generic reasoning systems that may coalesce the diversity of methods and techniques some time in the future.

Emerging Problem Areas: As we move toward the new millennium, the problems companies seek to solve through automation include organizational reengineering, work flow design, and yield management. A few innovative companies also seek competitive advantage maximization through information technologies.

Organizational Reengineering. Organizational reengineering is a form of organizational design. First, processes and information are modeled and matched to the corporate goals and mission. Anything not contributing to the mission is eliminated. The analysis techniques for performing the modeling and matching are a natural extension of enterprise analysis techniques from information engineering (Conger, 1994).

Remaining work processes and information are next analyzed to design jobs that can maximize quality and quantity of work products. Job design theory of Hackman and Oldham (1980) to do whole processes, make meaningful decisions, exercise discretion, and contribute to the organizational mission are followed. Then, using dependency and coordination theories (Galbraith, 1967; Thompson, 1967), jobs are grouped to maximize group cohesion and focus and to minimize inter-group linkages. The resulting organization is analyzed to determine effective enterprise-

wide potential for information technology to support the redesigned organization. The net result is identification of new and modified applications to support the reengineered organization. Thus, reengineering integrates systems analysis and organization theories in designing the new organization and its support systems.

Work Flow Redesign. Not all companies can cope with the radical change that accompanies a reengineering project. A scaled-down version of reengineering is work flow redesign. In work flow design, some particular work flow of an organization is analyzed for redesign. Paper flow through an insurance company is a good example. Policy and claims processing usually require the worker to have all policy related documents and information available to effect a change. In manual companies, this means a massive assembly, movement, and disassembly process for documents. The more people who touch a policy or claim, the more assembly, movement and disassembly. The process is error prone -- lost papers are common and may not be known to be missing for years. The process is time-consuming -- assembly movement and disassembly can take from four hours per policy/claim to indefinite time if problems in locating documents arise. The process is labor-intensive and necessarily sequential.

In work flow design, systems analysis techniques are used to identify the essential activities and skills and information as in reengineering. Then, both the jobs and computer systems supporting the jobs are redesigned to increase quality of service and quantity of output. The resulting applications might include mathematical application components to support, for instance, actuarial work; might include expert systems components to guide service representatives through, for instance, a retirement counseling session; and might include traditional transaction processing capabilities to guide automated image storage, retrieval, movement and tracking. Thus, work flow redesign combines and integrates the three problem types and methodologies into a new form of comprehensive application that supports parallel work of multiple people.

Yield Management. Yield management is undertaken to get the greatest return on a dollar of capital invested. Yield management deals

with optimal pricing of a limited number of products (e.g., rooms) when there are different product/price mixes (e.g., ski weekends, conventions, peak/low season), and when the demand for each product/price combination fluctuates and differs by location over time. Yield management, then, contains elements of mathematical and AI problems, with the solution based on data provided by transactional applications.

In the hotel industry, rooms wear out and must be refurbished on a fairly regular schedule, but rates charged for the room differ depending on season, lead time to rental, number of nights stay, promotions, and so on. Similarly, airline industry management involves pricing perishable airline seats at rates that will maximize revenues. Yield management in the manufacturing and hospitality industries is merging operations research methods with transaction and artificial intelligence methods to develop suggested rate changes throughout a given day or any other period of time.

The common element to all these emerging areas of systems analysis application is the need to integrate different theories and methodologies to develop solutions to these problems. This is a major change from the past 40 years during which the three problem types have been treated as distinct and separate problem-solving activities.

Support Mechanisms: Several types of support mechanisms are useful to facilitating systems analysis in the development of applications. Organizational supports include data administration and joint application design. Automated support is provided by computer aided software engineering (CASE) tools.

Data Administration. Data administration is the management of data to support and foster data sharing across multiple divisions, and to facilitate the development of database applications (Conger, 1994). Data administration organizations develop and maintain a data architecture for the organization which depicts the structure and relationships of major data entities, such as customers. The data architecture identifies automated and nonautomated data and how they are used in the organization. Users, working with data administration, develop an 'organizational' definition for each item. The architecture provides a

framework for defining new applications and documents all data uses and responsibilities for existing applications. Also at the organization level, users and data administrators work to define data that is critical to the organization as a going concern. Critical data is subject to management, standards, audits, security, and recovery planning. Noncritical data, while useful, is not required in event of disaster and does not require the same managerial attention (Conger, 1994).

At a lower level of detail, data administrators develop, administer, and maintain policies and standards for data definition, sharing, acquisition, integrity, security, access and so on for the corporation's data resource. Data administration provides guidance to project teams on storage, access, use, disposition, and standardization of data (Conger, 1994).

Some benefits from data administration are improved communication and understanding of corporate data from formal recognition and agreement on entity definitions, business rules and relationships. Application development efficiencies also occur because normalization of data takes place across the organization and not just at the individual application level. Redundancy becomes planned, and, therefore, can be managed. Organizations can react to changing business conditions faster because fully specified data definitions exist and can be used in new applications easily making control of critical data possible.

Data administration should not be confused with database administration which is responsible for physical database design, disk space allocation, and day to day operations support for actual databases. Instead, data administration optimizes data redundancy, provides shared understanding of data definitions, and is a managed approach for planning future database environments.

Joint Application Development. Several techniques have been developed to describe intensive user-analyst sessions during which requirements, designs, or other application related work is accomplished. The most common names are joint application development (JAD), joint requirements planning (JRP), joint application requirements (JAR), and

Fast-Track. (JAD and JRP are design techniques of the IBM Corporation. Fast-Track is a design technique of the Boeing Computer Services Company.) They are all similar in their goal of collaborative, user-analyst definition of application requirements. The planning and execution of joint sessions are also similar. The primary differences are the level of participants, subject matter, and level of detail of the discussions.

JAD is a team-based form of systems analysis which was designed to shorten the application development process and improve the quality of resulting application deliverable products. Some statistics show a 40% reduction in analysis time which leads to a faster implementation time. Other benefits of JADs are qualitative. Users and technicians develop a shared mental model and gain commitment to the joint work effort. Ideally, the spirit of teamwork continues into the other development stages.

A JAD team is composed of client representatives, a facilitator, systems representatives, and support personnel. The clients include decision makers at a level sufficient to resolve conflicts and make decisions that affect the scope and content of the application. They must also be at a low enough level to explain the daily functions and procedures of the organization. Clients from every functional area affected by the work should be represented. Systems representatives include the project manager, one or two senior analysts with technical expertise. The main role of the system representatives is to learn the problem area during the sessions to ensure accurate translation of the requirements in systems terms. In addition, systems representatives assess the feasibility and expected complexity of requirements for the target environment. The facilitator is a specially trained individual who runs the sessions, eliciting information, keeping the discussions on the topic, keeping the meeting moving, and identifying and resolving conflicts.

Both users and analysts prepare for the JAD sessions by attending training sessions in data and process modeling. Users are asked to prepare for the JAD session by gathering data examples and attempting to define the processes used in their own work. These user definitions become the basis of work the first day.

JAD sessions are three to five business days (or longer) at a site away from the normal work location. Daytime sessions are used to identify requirements, define terms, confirm business functions, identify constraints on processes and data, and so on. Evening sessions are used to document and circulate all work completed during the day for review the next morning.

The non-technical benefits of JAD are at least as important as the technical benefits. Users and analysts become friends and committed to a joint effort. Users and analysts develop a shared mental model that should be faithfully translated into the desired application. Staff time spent on analysis activities is reduced due to the intense, one-time session.

Computer Aided Software Engineering. Computer aided software engineering (CASE) tools automate aspects of the software engineering discipline. All CASE tools have a repository in which all design objects are defined and referenced. The more sophisticated the product, the more sophisticated the repository. The three distinguishing characteristics of CASE tools discussed here are level of the life cycle supported, level of intelligence imbedded in the software, and level of coupling to a methodology.

CASE products that automate analysis are referred to as *front-end CASE* or *Upper CASE*. CASE tools that automate program coding and testing are referred to as *back-end CASE* or *Lower CASE*. CASE products that are referred to as *I-CASE*, meaning integrated CASE, support multiple phases of work and integrate the work of one phase with that of succeeding phases.

In general, there are three levels of sophistication in CASE tools: dumb, semi-intelligent, and intelligent (see Figure 7). A dumb CASE tool (e.g., Briefcase, from Briefcase Corp.) is essentially a paper and pencil substitute that accepts information for storage and outputs reports on its repository contents. A semi-intelligent CASE tool (e.g., Visual Analyst, from Visible Systems Corp.) is one in which intelligent consistency and completeness checking are performed within a diagram or set of like definitions, but not across sets. An intelligent CASE tool (e.g., IEF, from

Texas Instruments, Inc.) provides not only intra-diagram set consistency and completeness checking, but also inter-diagram set consistency and completeness. The more intelligent the tool, the more checking is performed. IEF, for instance, is an intelligent tool that also analyzes impacts of changes on process and data definitions, generates third normal form relational database schemas, analyzes implementation feasibility of definitions, and generates error-free COBOL code with imbedded DB2 database processing.

The third distinguishing characteristic of CASE discussed here is coupling to a methodology. Coupling and intelligence are linked concepts in that the more closely coupled a CASE tool is to a methodology, the more the methodology's rules about definitions of graphics and design elements can be defined and enforced by the CASE tool. Tight coupling also has drawbacks. Tightly coupled CASE tools are inflexible to definitions that are not in the delivered product and sometimes require definition of graphics or items that are not required in a particular application. Both of these drawbacks can lead to extra work.

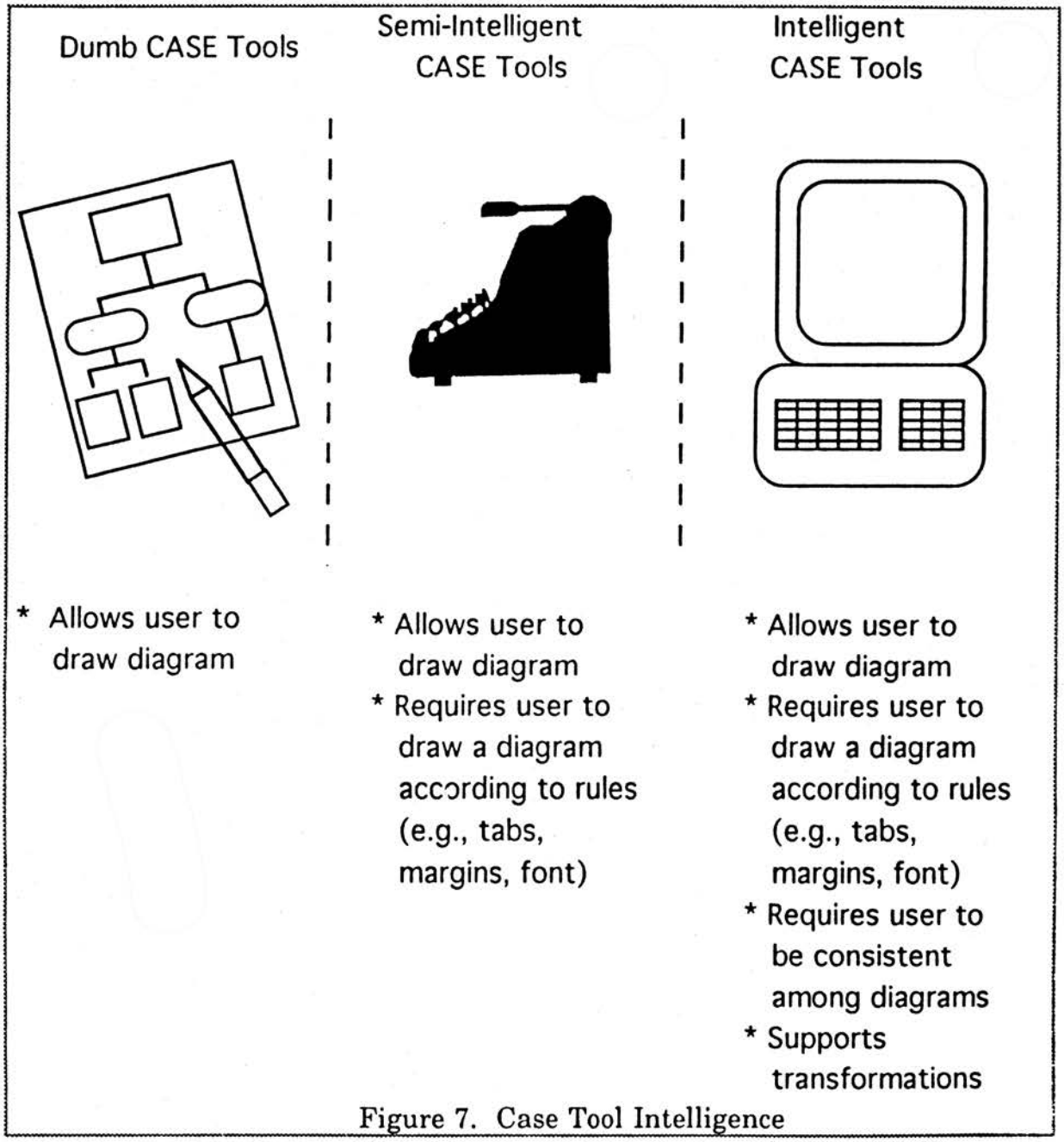


Figure 7. Case Tool Intelligence

Bibliography

Ackoff, Russell L., and Rivett, Patrick (1963). *A Manager's Guide to Operations Research*. NY: John Wiley & Sons.

Anderson, J. R. (Ed.). (1987). *Cognitive Skills and Their Acquisition*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Barr, A., and Feigenbaum, Edward (1981). *The Handbook of Artificial Intelligence*. Los Altos, CA: William Kaufmann, Inc.

Booch, Grady. (1987). *Software Engineering with Ada* (2 ed.). Menlo Park, CA: Benjamin/Cummings Publishing Co., Inc.

Booch, Grady. (1991). *Object Oriented Design with Applications*. Redwood City, CA: Benjamin/Cummings Publishing Co., Inc.

Buchanan, B., and Feigenbaum, E. (1981). "DENDRAL and Meta DENDRAL: Their application dimension". *Artificial Intelligence*, 11(1), 5-24.

Chen, Peter P.-S. (1981). "A Preliminary framework for entity-relationship models". In P. P.-S. Chen (Ed.), *Entity-Relationship Approach to Information Modeling and Analysis*. Saugus, CA: ER Institute.

Churchman, C. West, Ackoff, Russell L., and Arnoff, E. Leonard (1957). *Introduction to Operations Research*. NY: John Wiley and Sons.

Churchman, C. West (1968). *The Systems Approach*. NY: Delta (Dell) Publishing Co.

Churchman, C. West (1971). *The Design of Inquiring Systems*. NY: Basic Books.

Coad, Peter, and Yourdon, Edward (1990). *Object Oriented Analysis* (Second ed.). Englewood Cliffs, NJ: Prentice-Hall.

Coad, Peter, and Yourdon, Edward (1991).*Object Oriented Design*. Englewood Cliffs, NJ: Prentice-Hall.

Codd, Edgar F. (1972). "A relational model of data for large shared data banks". *Communications of the ACM*, 13(6), pp. 377-387.

Conger, Sue. (1994). *The New Software Engineering*. Belmont, CA: Wadsworth Publishing Company.

Dantzig, G. (1963). *Linear Programming and Extensions*. Princeton, NJ: Princeton University Press.

Date, Christopher J. (1990). *An Introduction to Database Systems* (Fifth ed.). Reading, MA: Addison-Wesley.

DeMarco, Tom (1979). *Structured Analysis*. NY: Yourdon Press.

Feigenbaum, Edward, Buchanan, B., and Lederberg, J. (1971). "On generality and problem solving". In B. Beltzer and Michie, D. (Ed.), *Machine Intelligence* (pp. 165-190). NY: Elsevier.

Hackman, J. R., and Oldham, Gregory R. (1980). *Work Redesign*, Reading, MA: Addison-Wesley.

Kent, William (1983). "A simple guide to five normal forms in relational database theory". *Communications of the ACM*, 26(2), pp. 120-125.

Kim, Won, and Lochovsky, Frederick H. (Ed.). (1989). *Object-oriented Concepts, Databases, and Applications*. NY: ACM Press.

Kolodner, Janet L., and Riesbeck, Christopher H. (Ed.). (1986). *Experience, Memory and Reasoning*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Laszlo, E. (1972). *The Systems View of the World*. NY: John Wiley and Sons.

Martin, James, and Finkelstein, Clive (1981). *Information Engineering*. Englewood Cliffs, NJ: Prentice-Hall.

Minsky, Marvin (Ed.). (1968). *Semantic Information Processing*. Cambridge, MA: MIT Press.

Papert, Seymour (1980). *Mind-Storms: Children, Computers, and Powerful Ideas*. NY: Basic Books.

Schlaer, S, and Mellor, Stephen J. (1992). *Object Lifecycles: Modeling the World in States*. Englewood Cliffs, NJ: Yourdon Press.

Smith, D., Buchanan, B., Engelmore, R., Adlercreutz, J., and Djerassi, C. (1973). "Application of artificial intelligence for chemical inference IX". *Journal of American Chemical Society*, 95, 6078.

Wagner, Harvey M. (1975). *Principles of Operations Research* (Second ed.). Englewood Cliffs, NJ: Prentice-Hall, Inc.

Ward, Paul T., and Mellor, Stephen J. (1985, 1986). *Structured Development for Real-Time Systems*. NY: Yourdon Press.

Warnier, J. D. (1981). *The Logical Construction of Systems*. NY: Van Nostrand Reinhold.

Weiner, Norbert (1948). *Cybernetics, or Control and Communication in the Animal and the Machine*. NY: John Wiley and Sons.

Winograd, Terry, and Flores, Fernando (1986). *Understanding Computers and Cognition*. Norwood, NJ: Ablex Publishing Corporation.

Yourdon, Edward, and Constantine, Larry L. (1979). *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Zahedi, Fatimeh (1993). *Intelligent Systems for Business: Expert Systems with Neural Networks*. Belmont, CA: Wadsworth Publishing Company.

Note: The following is a partial list of papers that are currently available in the Edwin L. Cox School of Business Working Paper Series. When requesting a paper, please include the Working Paper number as well as the title and author(s), and enclose payment of $2.50 per copy made payable to SMU. A complete list is available upon request from:

Business Information Center
Edwin L. Cox School of Business
Southern Methodist University
Dallas, Texas 75275

90-0101 "Organizational Subcultures in a Soft Bureaucracy: Resistance Behind the Myth and Facade of an Official Culture," by John M. Jermier, John W. Slocum, Jr., Louis W. Fry, and Jeannie Gaines

90-0201 "Global Strategy and Reward Systems: The Key Roles of Management Development and Corporate Culture," by David Lei, John W. Slocum, Jr., and Robert W. Slater

90-0701 "Multiple Niche Competition - The Strategic Use of CIM Technology," by David Lei and Joel D. Goldhar

90-1001 "Global Strategic Alliances," by David Lei and John W. Slocum, Jr.

90-1002 "A Theoretical Model of Household Coupon Usage Behavior And Empirical Test," by Ambuj Jain and Arun K. Jain

90-1003 "Household's Coupon Usage Behavior: Influence of In-Store Search," by Arun K. Jain and Ambuj Jain

90-1201 "Organization Designs for Global Strategic Alliances," by John W. Slocum, Jr. and David Lei

91-0101 "Option-like Properties of Organizational Claims: Tracing the Process of Multinational Exploration," by Dileep Hurry

91-0701 "A Review of the Use and Effects of Comparative Advertising," by Thomas E. Barry

91-0901 "Global Expansion and the Acquisition Option: The Process of Japanese Takeover Strategy in the United States," by Dileep Hurry

91-0902 "Designing Global Strategic Alliances: Integration of Cultural and Economic Factors," by John W. Slocum, Jr. and David Lei

91-1001 "The Components of the Change in Reserve Value: New Evidence on SFAS No. 69," by Mimi L. Alciatore

91-1002 "Asset Returns, Volatility and the Output Side," by G. Sharathchandra

91-1201 "Pursuing Product Modifications and New Products: The Role of Organizational Control Mechanisms in Implementing Innovational Strategies in the Pharmaceutical Industry," by Laura B. Cardinal

92-0101 "Management Practices in Learning Organizations," by Michael McGill, John W. Slocum, Jr., and David Lei

92-0301 "The Determinants of LBO Activity: Free Cash Flow Vs. Financial Distress Costs," by Tim Opler

92-0302 "A Model of Supplier Responses to Just-In-Time Delivery Requirements," by John R. Grout and David P. Christy

92-0303 "An Inventory Model of Incentives for On-Time Delivery in Just-In-Time Purchasing Contracts," by John R. Grout and David P. Christy

92-0304 "The Effect of Early Resolution of Uncertainty on Asset Prices: A Dichotomy into Market and Non-Market Information," by G. Sharathchandra and Rex Thompson

92-0305 "Conditional Tests of a Signalling Hypothesis: The Case of Fixed Versus Adjustable Rate Debt," by Jose Guedes and Rex Thompson

92-0306 "Tax-Loss-Selling and Closed-End Stock Funds," by John W. Peavy III

92-0401 "Hostile Takeovers and Intangible Resources: An Empirical Investigation," by Tim C. Opler

92-0402 "Morality and Models," by Richard O. Mason

92-0501 "Global Outsourcing of Information Processing Services," by Uday M. Apte and Richard O. Mason

92-0502 "Improving Claims Operations: A Model-Based Approach," by Uday M. Apte, Richard A. Cavaliere, and G. G. Hegde

92-0503 "Corporate Restructuring and The Consolidation of U.S. Industry," by Julia Liebeskind, Timothy C. Opler, and Donald E. Hatfield

92-0601 "Catalog Forecasting System: A Graphics-Based Decision Support System," by David V. Evans and Uday M. Apte

92-0701 "Interest Rate Swaps: A Bargaining Game Solution," by Uday Apte and Prafulla G. Nabar

92-0702 "The Causes of Corporate Refocusing," by Julia Liebeskind and Tim C. Opler

92-0801 "Job Performance and Attitudes of Disengagement Stage Salespeople Who Are About to Retire," by William L. Cron, Ellen F. Jackofsky, and John W. Slocum, Jr.

92-0901 "Global Strategy, Alliances and Initiative," by David Lei and John W. Slocum, Jr.

92-0902 "What's Wrong with the Treadway Commission Report? Experimental Analyses of the Effects of Personal Values and Codes of Conduct on Fraudulent Financial Reporting," by Arthur P. Brief, Janet M. Dukerich, Paul R. Brown and Joan F. Brett

92-0903 "Testing Whether Predatory Commitments are Credible," by John R. Lott, Jr. and Tim C. Opler

92-0904 "Dow Corning and the Silicone Implant Controversy," by Zarina S. F. Lam and Dileep Hurry

92-0905 "The Strategic Value of Leverage: An Exploratory Study," by Jose C. Guedes and Tim C. Opler

92-1101 "Decision Model for Planning of Regional Industrial Programs," by Uday M. Apte

92-1102 "Understanding the Linkage between Strategic Planning and Firm Performance: A Synthesis of more than Two Decades of Research," by C. Chet Miller and Laura B. Cardinal

92-1201 "Global Disaggregation of Information-Intensive Services," by Uday M. Apte and Richard O. Mason

93-0101 "Cost and Cycle Time Reduction in Service Industry: A Field Study of Insurance Claims Operation," by Uday M. Apte and G. G. Hegde

93-0301 "A Robust, Exact Alborithm for the Maximal Set Covering Problem," by Brian T. Downs and Jeffrey D. Camm

93-0501 "The Economic Dependency of Work: Testing the Moderating Effects of Financial Requirements on the Relationship between Organizational Commitment and Work Attitudes and Behavior," by Joan F. Brett, William L. Cron, and John W. Slocum, Jr.

93-0502 "Unlearning the Organization," by Michael McGill and John W. Slocum, Jr.

93-0503 "The Determinants of Corporate Bank Borrowing," by Linda Hooks and Tim C. Opler

93-0504 "Corporate Diversification and Innovative Efficiency: An Empirical Study," by Laura B. Cardinal and Tim C. Opler

93-0505 "The Indirect Costs of Financial Distress," by Tim C. Opler and Sheridan Titman

93-0601 "A Mathematical Programming Method for Generating Alternative Managerial Performance Goals After Data Envelopment Analysis," by Jeffrey D. Camm and Brian T. Downs

93-0602 "Empirical Methods in Corporate Finance used to Conduct Event Studies," by Rex Thompson

93-0801 "A Simple Method to Adjust Exponential Smoothing Forecasts for Trend and Seasonality," by Marion G. Sobol and Jim Collins

93-0901 "Leveraged Buyouts in the Late Eighties: How Bad Were They?" by Jean Helwege and Tim C. Opler

93-0902 "Stock Market Returns and Real Activity: International Evidence," by Thomas C. Harris and Tim C. Opler

93-0914 "Quality Management at Kentucky Fried Chicken," by Uday M. Apte and Charles C. Reynolds

93-0915 "Global Disaggregation of Information-Intensive Services," by Uday M. Apte and Richard O. Mason

94-0101 "Financial Distress and Corporate Performance," by Tim C. Opler and Sheridan Titman

94-0102 "Models of Incentive Contracts for Just-in-Time Delivery," by John R. Grout

94-0103 "Economic Dependency on Work: A Moderator of the Relationship between Organizational Commitment and Performance," by Joan F. Brett, William L. Cron and John W. Slocum, Jr.

94-0201 "The Antecedents of Block Share Purchases," by Jennifer E. Bethel, Julia Porter Liebeskind, and Tim Opler

94-0202 "The New Learning Strategy: Anytime, Anything, Anywhere," by John W. Slocum, Jr., Michael McGill, and David T. Lei

94-0401 "Leading Learning," by Michael E. McGill and John W. Slocum, Jr.