


1990

Management Information Sources and Corporate Intelligence Systems

Robert F. Gordon Ph.D.
Molloy College, rfgordon@molloy.edu

Follow this and additional works at: https://digitalcommons.molloy.edu/mathcomp_fac

 Part of the [Graphics and Human Computer Interfaces Commons](#), [Other Computer Sciences Commons](#), and the [Partial Differential Equations Commons](#)
DigitalCommons@Molloy Feedback

Recommended Citation

Gordon, Robert F. Ph.D., "Management Information Sources and Corporate Intelligence Systems" (1990). *Faculty Works: Mathematics & Computer Studies*. 15.
https://digitalcommons.molloy.edu/mathcomp_fac/15

This Book Chapter is brought to you for free and open access by DigitalCommons@Molloy. It has been accepted for inclusion in Faculty Works: Mathematics & Computer Studies by an authorized administrator of DigitalCommons@Molloy. For more information, please contact tochter@molloy.edu, thasin@molloy.edu.

Management Information Sources and Corporate Intelligence Systems

Robert F. Gordon

In this book the word "intelligence" is used in several different contexts. Intelligence can refer to the process of gathering data; it can refer to the data itself; and it can refer to the application of knowledge to produce useful information from the data. We will see in this chapter how the computer can be used in business to further all three aspects of intelligence: capturing the data, storing the data in an accessible form, and adding value to the data by transforming it into useful information for decision making. This chapter is organized according to these three areas of computer support for business intelligence:

1. Transaction processing and intelligence capture
2. Data-base management and intelligence storage and retrieval
3. Decision support systems and intelligence processing

We provide an overview of the concepts in transaction processing, data-base management, and decision support systems. References are listed in each of these areas for further details. Our purpose is to provide the perspective for the executive to determine the use of these concepts for his or her company and to understand the choices open to him or her in today's technology.

In discussing the application of computer systems to the intelligence function, it is important to distinguish between "data" and "information." The term "data" is used to refer to raw facts, such as individual sales transactions, bank transactions, or employee time cards. On the other hand, the term "information"

is used to denote the result of processing these facts in a meaningful way to lead to a decision. With respect to the given examples of data, the corresponding information could be the percent change in sales this year compared to last, the volume of bank transactions from each automated teller machine, or the number of employees late by day of the week. This information might help management decide, respectively, on marketing strategies for a product, the distribution of automated teller machines, and personnel policy.

Data and information are relative terms in that the material changes from one type to the other based on the current need of the individual in the company. What is information for someone at the level of operations control (such as hourly call demand to a reservations center) may be just data that needs further processing to be useful to higher levels of management. In this case the further processing might be to convert the hourly call demand into a schedule of operators for the supervisor by incorporating service-time information and union requirements. Middle management might require further processing to convert the hourly call demand to historical monthly demand and then to monthly forecasts that would then be converted to manpower projections for the next year. Here data external to the company, related to the industry and the economy, would be used to produce the forecast.

The conversion of data to information thus goes through several levels of information. It can involve accumulating or sorting the data, but more often the conversion requires incorporating other data as well as logic and mathematical algorithms.

Value is added when data is transformed into information, added again through each level of information, and added again when the information is transformed into decisions. The company gains value from transforming the transaction data of individual calls into hourly call demand to schedule the reservation agents, then into monthly demand (accumulation) and into monthly forecasts (external data capture and mathematical algorithms) to determine future requirements in staffing and equipment at the reservations center.

These value-added transformations are accomplished sometimes by the computer, sometimes by humans, and often by a combination of both. In the human/computer process, humans initially supply the objectives, criteria, and the process logic to the computer. Such specifications determine the software to be run by the computer. Once the software is programmed and tested, the computer provides the fast, accurate calculations, the execution of the specified algorithms, the storage and access of data, and the display of the resulting information. The human often interacts with the executing software, providing inputs (data and commands), viewing the results or partial results, and then evaluating them. This may lead him or her to change the input (what-if questions) or perhaps the algorithm, or perhaps to override the results. The human serves as the control when he or she makes these changes, and at times he or she may be able to program this control into the computer so it can be self-adapting. Iteratively and interactively the human/computer process leads to information and to decisions.

We call this process computer-based information systems and describe it in the next three sections.

TRANSACTION PROCESSING AND INTELLIGENCE CAPTURE

First, we discuss the input component of computer-based information systems. Transaction processing involves the capture, error checking, and updating of data.

A transaction is a grouping of data items (also referred to as fields). The data items define the various attributes of some entity. For example, an employee transaction is a grouping of the data items associated with a particular employee, such as employee number, last name, first name, address, department, and salary. An employee transaction would consist of these six data items. In determining the data items, we look at the requirements for the data and then set up each data item so that it is at the lowest level, that is, it does not have to be subdivided. Therefore, if we anticipate a need to access employees by city, address should not be a single data item, as in the example, but should be composed of several data items, perhaps street, city, state, and zip code.

The purpose of a transaction is to make a change to the data stored in the computer. This change can be an addition, a modification, or a deletion of data. Thus an employee transaction might add a new employee, change the pay rate of an existing employee, or remove a transferred employee from the employee data base. For now we will call data stored in a computer a "data base" and will define this term in the next section. Other examples of transactions are reservations, deposits, withdrawals, customer orders, and purchase orders.

Companies can enter transactions into a computer in several ways. We identify variations in terms of hardware as well as processing methods. In terms of hardware, companies often utilize other computers, terminals, or sensors at remote sites to capture data. Communications networks are used to transmit data between terminal and computer, as well as for computer-to-computer links. A company may retrieve the data from another site to store and process in its own computer, or it may use the processing power of another computer to both extract and manipulate the information needed.

In terms of processing methods, companies may use interactive processing, batch processing, or a combination of both. Interactive processing allows the user to enter and edit individual transactions, getting immediate feedback during the process. Batch processing allows for bulk updates with feedback after all transactions are processed.

In interactive processing, transactions are entered at a terminal. Typically a program displays prompts and information on a terminal, and the operator at the terminal enters replies and commands. The operator interacts with the program in an iterative loop of entering data and getting feedback (information). The program allows the operator to edit the responses and transmits the operator's input to the computer, where it can be checked and perhaps, after several iter-

ations of input and response, used to update the data base. Based on the operator's replies and/or the result of the update, the program displays the appropriate prompts and information back to the terminal.

As an example, a customer may telephone a hotel reservations center and provide the reservations agent with the date items for a reservation transaction. The agent utilizing a software program is prompted first for the type of reservations transaction: make a reservation, modify a reservation, delete a reservation. If the agent selects "make a reservation," the program displays a template, prompting for the customer's name, hotel, expected arrival date and time, expected departure date and time, room type, number of people in the party, credit card type and number, and rate code. When completed by the operator, this transaction is transmitted to the computer, where availability at that hotel for those dates and room type can be checked. If a room is available, the transaction can update the reservations data base, and a confirmation message can be sent back to the agent at the terminal. Thus the reservation can be made and confirmed while the customer is on the line.

In addition to fast feedback, another benefit of interactive processing is that previously stored data can be used to guide the input. Thus, if a customer calls to modify a reservation, the reservations agent enters the customer's identification, such as a reservation number. When the agent enters this identification, the data corresponding to that reservation are displayed, serving as verification and prompt information for talking with the customer and as a basis for modifying the reservation. The agent avoids having to ask the customer and reenter previously stored data.

A similar interaction occurs when a terminal is used to enter customer orders or to record shipments or payments. As the data is received, it is entered, edited, and used to update the data base. There is relatively small delay between data entry and use in these types of systems; thus availability of rooms, inventory, and orders are kept up to date with the activity. Furthermore, interactive processing allows the corrections to the input to be made at the time of data entry, before the update of the data base.

Some business environments lend themselves to direct input of data by the customer, rather than having the customer go through an intermediary. Thus a bank customer might enter a withdrawal or deposit transaction directly into the terminal (automated teller machine) instead of giving it to the bank teller to enter. A traveler might reserve an airline seat by interacting with a touch-screen terminal that displays a diagram of the airplane's seating. A visitor might find a list of restaurants by selecting items from menus on a terminal screen or find travel directions by selecting one of a set of sites on a map displayed on the terminal screen. In the hotel reservations example, instead of the customer giving the reservations agent the data, he might use a touch-tone phone to input the information directly to the computer.

To further reduce the delay between an event and its entry into the computer, the data capture might be automated. Sensors can capture the data instantaneously

and send it to the computer, avoiding the time to communicate it to the terminal operator and to have it typed into the terminal. Applications of sensors are radar to transmit air traffic control programs, onboard instruments to capture positional data for missile guidance control, road sensors to control traffic lights, measurement probes sending data to control a chemical process, and optical scanners to read product bar codes on a shipping/receiving dock. The term "real-time" might be best applied to these systems; the computer data is updated as the event (transaction) takes place.

In other cases the data might already be in a form readable by computer but might be at another site or in another company's computer. A company with branch computers might store local data in each branch computer and connect the local computers through a communications network. This distributed processing allows local processing of data as well as the transmission of data to the other computers for processing and/or display.

News services, stock trading, government statistics, financial data, and economic projections often are put in computer-readable form. Companies can access this data for a fee through a telecommunications link to the service's computer. The service's computer provides the data and the access program. A company can extract selected data from a service's computer data base to be used in its own computer data base.

In contrast to the given examples of data entry, companies may want to use batch processing for some updating of their data bases. In batch processing all the transactions for a given period are converted to machine-readable form, such as on tape or disk. Usually an operator using a key-to-tape or key-to-disk machine enters and verifies the transaction data. The resulting tape or disk is read by a program that updates the whole data base with the transactions. This approach is best suited for a large number of transactions when most of the data base must be updated and there is no need for immediate response or prompting for input. Neither the computer nor the communications line is tied up for data entry as in interactive processing. Batch processing is typically used to update data bases to produce periodic reports. Most accounting applications, such as general ledger, accounts receivable, accounts payable, and payroll, are effectively handled in this way.

Getting the data into computer-readable form is often the most time-consuming and error-prone part of using a computer system. In addition to key-to-disk and key-to-tape entry, optical scanning and optical reading can be used to capture the data. Optical scanning programs identify the presence or absence of markings (whether penciled multiple-choice answers or bar codes). Optical reading, on the other hand, requires that the computer identify specific characters or numbers. Optical reading is thus more processor intensive than optical scanning.

Voice input is beginning to be used in some applications. Discrete speech recognition (recognizing speech with distinct pauses between words) is an alternative means of data entry. Programs exist that can recognize a large number of words. Most often these recognition programs are voice-dependent. They may

be useful, for example, on an assembly line where the operator's hands are not free to input data. Continuous speech recognition is much more difficult to accomplish by computer, requiring very large processing capability.

For some applications, it is not necessary that the computer recognize the content of the data, whether it be text, voice, graphics, or images, but only that the computer be able to store and retrieve it. Thus pages of text, voice, graphics, and images can be scanned and digitized to be stored (and/or transmitted) in a computer. Programs retrieve the items and present them to the user; it is left to the user to process them.

DATA-BASE MANAGEMENT AND INTELLIGENCE STORAGE AND RETRIEVAL

Once the data is captured for computer use, it is necessary to store it in a form that makes it easily accessible. Different storage methods and means of structuring the data are used to organize the data for fast retrieval. The particular storage method and data structure depend on the retrieval needs. We will describe several storage methods and data structures and discuss the conditions under which to use each. In doing so, we will concentrate on the important data-base concepts.

First, some terminology is necessary. We spoke about data items comprising a transaction in the previous section. It is common to refer to the data items as fields and transactions as a type of record and to think in terms of the following hierarchy: a file is made up of records that are made up of fields. Thus an employee file is composed of employee records (one for each employee), each of which is composed of fields (each field containing the value of a data item for that employee). Typically the format of each record in the file is the same. In this case each employee record has the same number, type, and size of fields. The first field of every employee record may be employee number, the second field may be employee last name, and so on. The format of the fields forms a template, more precisely called a record type, to be followed by each record. When the actual values for fields are supplied, such as employee number: 83067, employee last name: Jones, and so on, a record is created. Each record is an example or, more precisely, an instance of the record type. A file usually has one record type and contains many instances (one for each employee in the example) of that type. Usually one field of the record type is denoted as the key field; it is a field that can be used to order the records in the file. Typically a field that has a unique value for each record is chosen to be the key field. In the example, employee number would be a good candidate for the key field.

Just as an executive might have memos organized in a file folder, the computer has records organized in a file. The executive might have organized memos in one file folder in chronological order, in another file folder by subject, and in another by sender. Similarly, if we consider each memo to be a record and its date, subject, and sender to be three different key fields, then the computer can

be instructed to store the data in one of these key sequences. Given any of these keys, this could provide any of these organizations of the memos.

It is important to realize that it is not necessary to physically store the same data in three different ways (three files) in order to access it efficiently in each way. Furthermore, we may not want to store the data in any of these sequential orders, and yet we may want to be able to utilize the three orderings.

As an example, if the executive only stored the memos chronologically in one file folder but prepared a list (index) by subject showing the dates associated with each subject, he could use the index to point him to the appropriate memo in the chronological file, thereby accessing the memos in subject order. Of course this would be slower than having the file in subject order for this purpose, but this procedure eliminates the need to duplicate the memos. Similarly, indices and pointer fields allow the computer to access data in many different ways even though the data is stored only once.

The access method refers to how to find the record physically on the storage medium. The storage medium is usually tape or disk. The data structure of the records refers to the organization of the data: the order of fields within a record, the connection between records in the file, and the use of indices and pointers to reach a record.

Regardless of the organization of the data, there are two data access methods: sequential access and direct access. In sequential access, each record is retrieved in sequence as it is stored. A typical sequential access method device is a tape drive. The data structure for sequential access is usually also sequential; that is, the records are usually in some meaningful order (alphabetical, chronological, or the like). In direct access, any record can be retrieved without having to retrieve the prior records in the file. This method allows the computer to go directly to the address (location) of the desired record. The records need not be in any specific order required by the application; that is, the physical location of the records need not be determined by the logical relation of the records. A typical direct access method device is a disk drive.

Sequential access can be compared to a cassette player. To play the fifth song on the cassette or to edit it, you must play or fast-forward through the first four songs. In contrast, direct access is similar to a record player; you can play the fifth song directly by moving the record arm to that song. The listing of the songs on the album cover provides the index to each song on the record, and you can play them in any order.

To get an appreciation of the benefits and limitations of each method, consider the playing of one particular song in the album. Sequential access (cassette player) would require playing all prior songs; on the average, half the songs would be played before reaching the desired song. Direct access allows us to move the record arm to the selection without playing through the previous songs. In this case (specific selection), direct access is superior. On the other hand, suppose we wanted to play every song on the cassette. We would start the cassette and sit back and listen to each song in sequence until the cassette ended.

If we were to follow the direct access scheme, manually we would check the album index and pick up and place the record arm before each song. In this case (all selections), sequential access is better. Of course, with the phonograph record we could let the record arm start at the beginning of the record and play through all of the songs automatically. This points out that a direct access device, such as the record player, often can be used for both direct access and sequential access.

There are many ways (data structures) to organize and store company data. Which structure to use depends on the retrieval requirements for that data. In a sequential device there is basically only one type of data organization, and that is called a sequential file. In this data organization each record is sequenced by a key field. The records could be in alphabetical order if, for example, last name were the key field. The records could be in chronological order if date were the key field. Accounting applications often use sequential files because the application is run periodically, and for each period, most of the records in the file need to be accessed. For example, a payroll system will have to read the record for each employee to produce the paychecks each period. As we saw earlier, when most of the records need to be accessed, a sequential file structure is best.

Sometimes there are several different and possibly conflicting retrieval requirements for a given set of data. A limitation in the payroll sequential file structure is that if we wanted the payroll to be produced in employee number order and a payroll report to be ordered by employee last name, then we would have to store the same data twice, once for each order required. With a direct access method device, we can handle this and only store the data once. In the example of the record album, if we wanted to play all the songs but in a different order, we could use the index on the album cover to point us to move the record arm to each song in the desired order. This is true for computer direct access devices as well.

For a direct access method device, there are many choices of data organizations. This flexibility allows the design of the data structures to fit the multiple needs of the company. As we saw earlier, a sequential data structure can be handled on a direct access method device. For other data structures, the computer must be supplied with the location of the record desired. This can be provided by giving the computer access program the desired record's key and having the program transform the key algebraically to arrive at a number that will be assigned as its location. Such a technique is called basic direct access method (BDAM), and typically the key is divided by a prime and the remainder is the recorded location.

Another technique utilizes an index. The indexed random structure has an index stored with the records on the file. The index contains two columns (like the index of a book or album cover); the first column contains the key values in sequence, and the second column contains the corresponding location of that record. To retrieve a record, the computer program finds the desired key value in the index and retrieves the record from the corresponding location.

A third method utilizes a field in each record to point to the location of the

next record to process. Once the computer retrieves a record, one of its fields contains all the information necessary to retrieve the next record. Pointers in records avoid the need to use keys. With several pointer fields in a record, the computer can process the data in many different ways, although the data is stored only once. All that is necessary is to be given the field to use for the pointer and the initial record's location. Even parts of a record (segments) could be stored separately, as long as there is a pointer from one segment to another. This is the basis of a data base.

A data base is the data together with a description of how the data is organized. The description of the data organization is called the schema. A program, called a data-base management system, reads the schema and uses it, as one would use a road map, to find the data in the data base. The schema tells the data-base management system the location, or how to find the location through other records (using pointers), of the desired record or record segment. The files that we described previously (such as a payroll file) have one record type. These files also are organized in one way (for example, sequentially, one record after the other) and are called flat files. With pointers, the file can have a more complicated and more versatile organization. Its structures could have some records at a different level of importance than others or a different grouping of records.

The data-base management system provides the ability to specify the schema, load the data base (instances), update the data base, and retrieve information from the data base on the schema. It is this last function that is the purpose of setting up the data base and the one in which we are most interested. The purpose of a retrieval might be to produce a report from the data base by finding (and perhaps accumulating) all records that satisfy a particular criterion (request). Alternatively, we may want to interactively query the data base with an English-like language to display specific fields from all records that meet a criterion.

The two most popular kinds of data bases in industry are hierarchical data bases and relational data bases. A hierarchical data base is used when the data organization required for retrieval follows a natural hierarchy, similar to a company's organization chart. The schema for a hierarchical data base starts at the top of the organization (root segment) and provides pointers to each succeeding level. Thus, to use the personnel organization chart as an example, the root might be the president record (it might contain fields referring to name, office location, duties, and so on), the next level might be the vice president record (it might contain different types of fields, such as functional department name), the next level might be the director level, and so on. This schema provides an outline of the data in the data base. When the actual fields are completed, the resulting instances of the data make up the data base. There may be one instance of president, ten instances of vice president, and so on, just as there were many instances of the employee record type in the payroll flat file. This hierarchical data base is useful for retrieving all employees working for the marketing vice president, for example, because they are all in the levels pointed to by that vice president's record.

Applications that lend themselves to hierarchical retrieval are reporting sys-

tems: division, region, and territory for sales reporting or for financial statements. Thus, to retrieve a salesperson's activity, the data-base management system will select from the data base the appropriate division, which will point it to the appropriate region, which will point it to the appropriate territory, which will contain the salesperson's activity in one of its fields. This structure allows fast retrieval of all salespersons in a particular region. The reverse retrieval, to find a particular salesperson without knowing his or her division and/or region, would be difficult (time-consuming), because the hierarchical structure does not help the search.

Relational data-base organization is preferable when we want the capability to change the order of retrieval based on the request. For example, sometimes we may want to retrieve information about all the customers who bought a specific product; at other times we may want to get information about all the products bought by a particular customer. Customer and product form a different hierarchy in these two cases. In the first case it would be better for retrieval efficiency to have customer at a higher level. Then getting all the product records under a specific customer would be easy; however, with that organization it would be very time-consuming to determine all the customers for a given product.

A relational data base is designed to handle situations like this, where the hierarchy needs to be changed for different retrievals. A relation is simply a flat file or a table of records. A relational data base has one or more relations. The schema for a relational data base defines each relation or table by identifying the fields that each record in that table will contain and which field will be the key. The relational data-base management system provides functions to manipulate the tables, such as selecting the instances of a table that meet certain criteria for their fields (all employees with salary field above \$50,000), or joining instances from two tables that have a field value in common to form a third table.

In the example there could be three relations: product, customer, and purchases. The product relation might be a table of all product records, each with fields such as product code (key), price, and inventory amount: Similarly, the customer relation might be a table of all customer records, each with fields such as customer name (key), address, and balance. In order for the data-base management system to retrieve product information about all products purchased by a given customer or customer information for all customers who purchased a given product, we need to provide it with information from both relations. A purchase relation might contain all purchase records with fields such as customer name, product code, and amount.

To answer the request for all product information for products purchased by customer A, the program would first search the purchase relation for all instances of customer A and from the resulting product code in each one of these instances would retrieve from the product relation the information for that product. To get the customer information for all customers that purchased product 001, the program would first search the purchase relation for all instances of product 001

and from the resulting customer name in each one of these instances would retrieve from the customer relation the information for that customer.

Thus a relational data-base management system finds the necessary links between the relations as it processes the request. It is therefore more flexible than a hierarchical data base. The hierarchical data-base management system has the links already defined between the levels of the hierarchy. It is thus able to find the desired records faster than a relational data base, provided that the request is based on those predetermined links.

The data-base management system (whether hierarchical or relational or other) then allows management to query the data base and provides the results of a wide range of searches through the large number of instances in the data base. Management can get information in the form of reports or interactive answers to queries. The information is based on selection criteria and can consist of details of individuals as well as group summaries, subtotals, and comparisons. Basic arithmetical and logical combinations of the fields in the selected records can be produced for management decision making.

For example, personnel management might ask to select from their employee data the employees who have experience in certain areas, high ratings, and certain aptitudes as a basis for filing a job application. Marketing management, for advertising decisions, might query the reservations data base to calculate the mean lead time between reservation booking date and arrival date for a particular time of the year and region of the country. A customer service agent might ask for the billing information to be displayed for a particular customer to answer a complaint.

Typically the data stored in data bases are internal to the company, historical, and very detailed (at the individual transaction level). The data-base management system searches and combines the data to answer the query or to fill the report. This is useful to help solve the types of problems and information needs at the lower levels of the company, where the internal, historical, detailed information can be used as a basis for decisions. Specific employees, customers, or products can be selected that meet criteria. Summary and statistical information can be provided based on the detailed field values, such as lead-time information. Management has complete access to, and the ability to select and combine all of, the data captured in its data base.

Sometimes the extracted information completely answers the problem (list of employee candidates); sometimes further transformations are necessary to arrive at a decision (the lead time is just one input to determine the media plan). We will describe in the next section decision support systems to apply to the extracted information that evaluate alternatives and help management determine actions.

It is important to realize that top-management decisions usually involve long-term estimates that cross a wide number of functional areas in the company. These decisions require less detailed information than those at lower levels of the company; they can often be based on approximate and summary data. How-

ever, they usually require additional external data (economy, competition, government), as well as future-oriented data.

For example, to predict the next several years of company sales might require thirty-six numbers (the last thirty-six monthly sales figures) and economic and competitive action projections for the next several years. The individual detailed transactions that make up these thirty-six numbers are of little value in this case except in aggregate, so the thirty-six numbers might more practically be kept on a piece of paper (or duplicated in some other file) than calculated each time from the data base. Add to this that the external projection data are typically not even in the data base, and we see that in this case the data-base retrieval is not the important contribution of the computer; it is the decision methodology that is important. Given that these are the kinds of problems (long-term, future-oriented, requiring external data) that top management faces, top management benefits less from transaction processing and data-base management than the lower levels of the company.

Top management can, however, benefit as much as the other levels from the use of decision support systems. We will describe some of the most often used decision support systems in the next section and indicate the type of real-world problems that they should be used to solve.

DECISION SUPPORT SYSTEMS AND INTELLIGENCE PROCESSING

Decision support systems is a term used to describe a wide range of computerized solution techniques that can be applied to solve business problems. The solution techniques may describe alternative scenarios (answer what-if questions) or produce optimal solutions. Some may merely be data-base query routines, while others may incorporate complex mathematical algorithms. Some techniques may be developed for a company's specific problems; others may be general techniques that can be applied to similar kinds of problems in many companies.

We will describe at a high level three different and widely used techniques to give an idea of the type and range of methods available. For each technique we give a brief background of the concept and assumptions followed by an example of its use in business. In order to show a manual solution we simplify the real-world problem here, reducing its complexity in terms of the number of variables, number of outcomes and options, and number of time periods; the computer is required for most realistic business applications of these solution techniques, but the computer solution techniques follow the principles shown here.

Given a problem, the decision maker needs to determine what solution technique to use and then to abstract from the real world the important aspects of the problem required by that solution technique. The result of this abstraction is called a model.

The model is a representation of the real system, that is, a simplification of the real system, including only those aspects that are considered important to the problem at hand. The model might be a graphical representation of the problem to be solved (such as an architectural drawing of the building or an engineering drawing of a machine part), or it might be a mathematical representation (such as a profit function with production constraints or a queuing equation). The resulting model can be analyzed and revised to examine alternatives. The desired objective is that the best alternative be identified by using the model and then translated to the real system and implemented.

We can categorize business problems into decision making under certainty, decision making under risk, and decision making under uncertainty, based on how much information is known. Each of the first two categories has a set of solution techniques that have been found most useful in solving that category's problems. In the third category, decision making under uncertainty, there is insufficient data to make a decision without imposing some outside conditions or spending some money to get the additional data.

For decision making under certainty, it is assumed that the decision maker knows all the possible actions that can be taken, that for each action there is only one outcome, and that the payoff for that outcome is known. The problem is to determine the action, perhaps from an infinite number of possible actions, with the best payoff. Many problems in decision making under certainty in industry have been solved by applications of differential calculus or by mathematical programming. These techniques have been used to solve problems ranging from inventory management and distribution to production planning and scheduling.

For decision making under risk, it is assumed that the decision maker knows all the possible actions that can be taken and that for each action he knows all the possible outcomes with the payoff and probability of each. Two solution techniques for solving very different kinds of decision-making-under-risk problems are decision trees and simulation. Decision trees have been used to solve problems involving media selection, pricing, new-product introduction, and investment decisions. Simulation has been used to analyze performance-improvement decisions in areas ranging from transportation systems and communications networks to manufacturing lines.

For decision making under uncertainty, it is again assumed that the decision maker knows all the possible actions that can be taken and that for each action he knows all the possible outcomes, but he does not know the probability associated with each outcome. In decision making under uncertainty, the technique of payoff matrices is often used as a starting point.

In each category, when we say that the decision maker knows all the actions or outcomes or that he knows the payoffs or the probability, we mean realistically that he knows the important action choices and outcomes and that he can reasonably estimate the payoffs and probabilities in a range that is accurate enough to make the decision. We describe a popular solution method used on the com-

puter for solving problems in decision making under certainty and two methods for solving problems in decision making under risk.

Decision Making Under Certainty

In decision making under certainty, there are a number of optimization methods available for solving business problems by computer. We discuss here a special case of mathematical programming, called linear programming, that has been successfully used in optimizing the refinement of oil, the production of feed mix, and the manufacture of pharmaceuticals.

Linear Programming: Brief Background

Mathematical programming is a solution technique applied to decision problems where entities are competing for limited resources, whether those resources are labor, machines, storage, or demand. In mathematical programming the real system is modeled by an objective function that is to be optimized (either maximized or minimized) subject to a set of constraints. The general formulation of a mathematical program is to optimize

$$f(x_1, \dots, x_n)$$

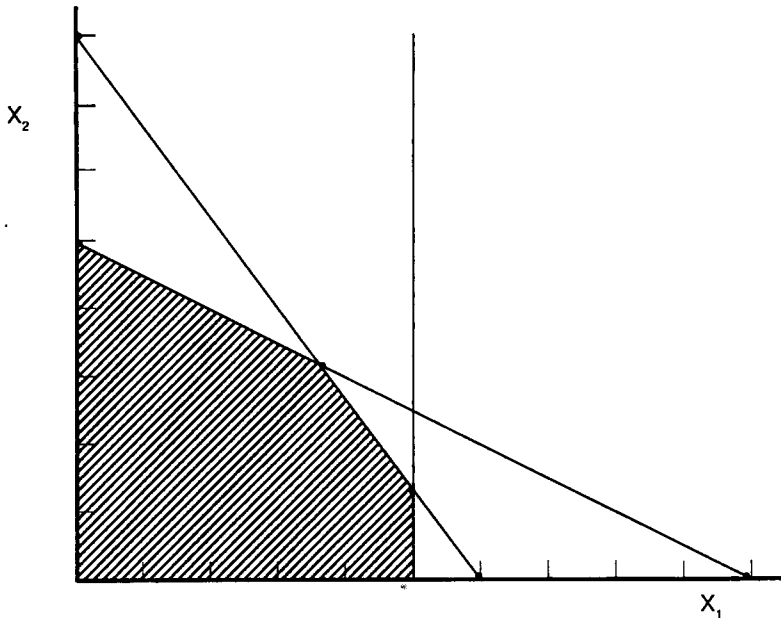
subject to

$$\begin{aligned} g_1(x_1, \dots, x_n) &\leq b_1 \\ g_m(x_1, \dots, x_n) &\leq b_m \end{aligned}$$

where $x_1, \dots, x_n \geq 0$. In this formulation, any constraint g_i can be less than or equal (as shown), equal, or greater than or equal to b_i . We want to find non-negative values of x_1, \dots, x_n that will optimize the function f while satisfying all the constraints g_1, \dots, g_m .

A linear function is one that is a sum of terms each of which is a constant times a variable raised to the first power. A linear function of two variables set equal to a constant is a straight line; a linear function of three variables set equal to a constant is a plane; a linear function of more than three variables set equal to a constant is a hyperplane. In particular, if both the objective function f and the constraint functions g_1, \dots, g_m are linear functions, then the model formulation is a linear program. A linear program with even a large number of variables and a large number of constraints can be quickly solved by computer (using the Simplex Method or some variation of it) to provide the optimal solution. The basic assumption to use this method is linearity of the objective function and the constraints; that is, a change of one unit in the value of a variable produces a constant change regardless of the value of the variable. Thus there can be no diminishing returns and no economies of scale.

Figure 3.1
Feasible Region



To get an idea of how the Simplex Method produces an optimal solution, consider the case of two variables and the following linear program: maximize

$$f(x_1, x_2) = 10x_1 + 15x_2$$

subject to

$$g_1(x_1, x_2) = 4x_1 + 8x_2 \leq 40$$

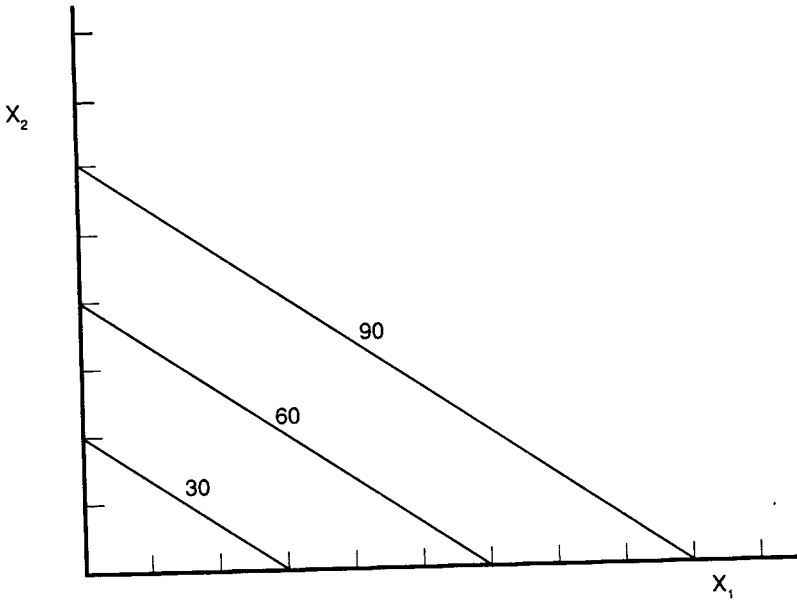
$$g_2(x_1, x_2) = 6x_1 + 4x_2 \leq 36$$

$$g_3(x_1, x_2) = 1x_1 + 0x_2 \leq 5,$$

where $x_1, \dots, x_n \geq 0$.

Figure 3.1 shows the region of x_1 and x_2 values that satisfy the three constraints (feasible region). We then want to determine which of this infinite number of points optimizes the objective function. Figure 3.2 shows some possible curves for the objective function obtained by setting $f(x_1, x_2)$ to some test values. All the possible objective function curves are parallel to each other, and their values increase as we move toward the top right of the figure. Overlaying the possible objective function curves of figure 3.2 on the feasible region of figure 3.1, we see that the optimal solution is at the point where the objective function last cuts

Figure 3.2
Objective Function Values



the feasible region as we move toward the top right. This occurs at the corner $x_1 = 4$, $x_2 = 3$ of the feasible region.

In the two-dimensional case, because the constraints are all linear, the feasible region is bounded by straight lines, and because the objective function is linear, the objective function is a straight line. If we move the objective function line until it last intersects the feasible region, then the optimal solution will always occur at a corner (or, in the case where the objective function is parallel to the boundary line, at the whole boundary line as well as its corners) of the feasible region. This is true whether we are maximizing the objective function or minimizing it and generalizes to any number of variables and any number of constraints. Since the number of corners in a feasible region (regardless of the number of variables and the number of constraints) is finite (whereas the number of points that make up the feasible region is infinite), an algorithm can solve the problem by testing the corners. The Simplex Method starts at one corner, tests the direction that will most improve the objective function, and moves to that corner. It continues this process until it reaches a corner where going in any direction can no longer improve the objective function.

Linear Programming: Example

Let us say that we have two products A and B that we can produce, and let x_1 represent the amount of product A to produce and x_2 represent the amount of

product B. Suppose that we will get a profit of \$10 per barrel of product A and \$15 per barrel of product B that we produce. Obviously there are constraints on how much we can produce. Suppose that the production of each product requires time on the same two machines and that machine 1 is available for 40 minutes per hour and machine 2 is available for 36 minutes per hour. Suppose that to produce a barrel of product A requires 4 minutes on machine 1 and 6 minutes on machine 2. To produce a barrel of product B requires 8 minutes on machine 1 and 4 minutes on machine 2. Suppose also that the number of barrels of product A must be less than 5 per hour because of demand limits. The resulting linear program representation of this problem is the one shown in the "Brief Background" section. The optimal solution is the corner point $x_1 = 4$ barrels of product A, $x_2 = 3$ barrels of product B, and the profit at that production level is \$85 per hour (see figures 3.1 and 3.2). There is no other combination level that will give a better profit.

Decision Making Under Risk

We describe the main aspects of decision trees and simulation as examples of two very different and widely used solution techniques for decision making under risk. Decision trees are usually used to make marketing decisions on price, advertising, and product introductions and financial decisions on investment alternatives. Simulation is often used to make production and staffing decisions.

Decision Trees: Brief Background

In decision making under risk, it is assumed that the decision maker knows all the possible actions, the possible outcomes for each action, and the associated probability and payoff of each action. The decision tree is a technique of diagramming these alternatives and applying a rule (expected value) to eliminate those actions that are not part of the optimal strategy.

The decision tree displays the possible actions for each time period (shown as branches emanating from an action point) and, for each action, the possible outcomes with their probabilities. Each path, consisting of alternating action and outcome branches for each period, results in a specific cumulative payoff. To solve the decision tree, we start at the last period and calculate the expected value of each action in that period. The expected value is the weighted sum of the probabilities and the cumulative payoffs:

$$\text{Expected Value} = p_1P_1 + \dots + p_nP_n$$

where p_i is the probability of outcome i , and P is the payoff for that branch of the tree.

The expected values of each possible action at an action point are compared and the best one selected; all other action branches are eliminated at that action point. This selected expected value is then used as the payoff in the calculation

of expected value for the prior period, and the process is repeated for the next-to-last period. The result (paths remaining) after completing all the time periods is the optimal strategy.

Inherent in this process is the assumption that the action with the best expected value is the best action. First of all, since the expected value is a weighted average of the payoffs, it is most likely the case that the actual payoff will not be equal to the expected value. The expected value of an action with two equally likely outcomes, one with payoff \$2,000 and the other with payoff \$1,000, is \$500. That payoff will never be obtained. If the action can be repeated over and over again under the same conditions (a rare situation in business), the long-run average payoff will be \$500. However, given that the action is to be made once, is that action better than one with a guaranteed payoff of \$400? In reality, the answer depends on your situation: whether you can or want to risk losing \$1,000; whether you have a requirement to choose only those actions that have a chance of returning at least \$2,000; and so on. Often utility is substituted for dollars in measuring payoffs to account for some of the limitations in using expected value.

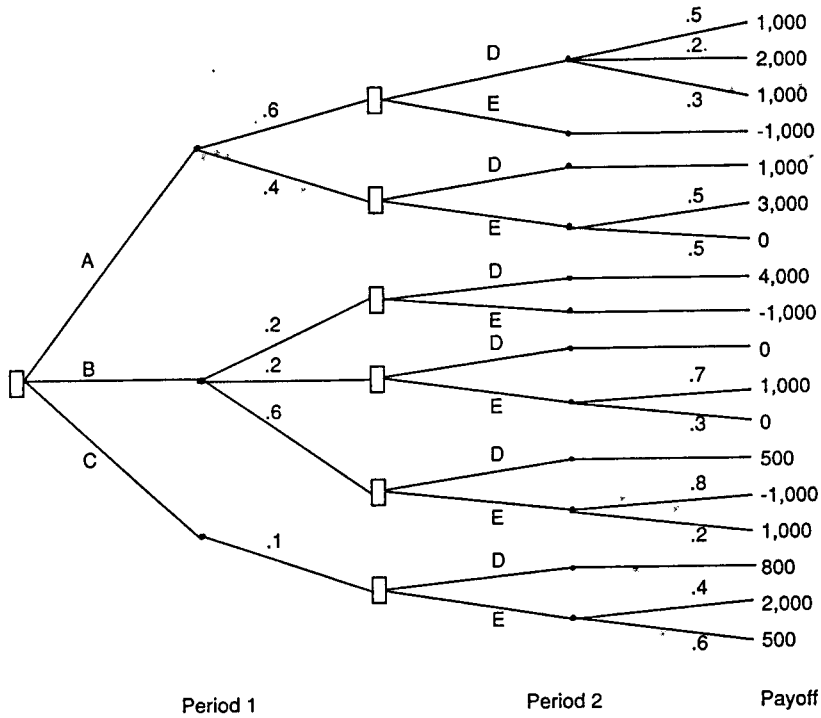
Decision Trees: Example

As an example, suppose that a company wants to determine an advertising strategy. Let us assume that it needs to decide between three advertising plans, A, B, and C, for the first month and D and E for the second month. Its estimates of the possible outcomes and the probabilities and the resulting two-month payoffs are shown in figure 3.3 for every realistic combination of action and outcome. We work backwards from period 2. At each point either action D or E is selected based on the higher expected value. The remaining action's expected value is used as the payoff for period 1. As a result, action A in period 1 gives the highest expected value, \$1,320. For action A, the expected value of D is better than E if outcome 1 results in period 1 and the reverse if outcome 2 results. The optimal strategy, therefore, is to choose A in period 1 and then if outcome 1 results, choose D in period 2; otherwise choose E. We point out again that given the assumptions, there is no better strategy. Note, however, that the results of this strategy could never produce the highest possible payoff (\$4,000) as with action B, nor guarantee a positive payoff as in the case of action C.

Simulation: Brief Background

Simulation is a solution technique that mimics the real system by describing the activity flow in a model. A discrete event simulation program generates an artificial flow of customers in the model, tracks them as it advances time in the model, and determines the resulting performance of the system in terms of measures such as delay time and waiting-line lengths. The model can then be changed by revising conditions, such as adding more servers, changing the arrival pattern, or changing the service rules. The resulting performance measures can then be compared to see which combination provides the desired performance at a reasonable cost.

Figure 3.3
Decision Tree



Source: John A. Campbell, James R. Hemsley, David Burris Windsor, *Artificial Intelligence and Management*, The Institute of Management Sciences, August 1986, p. 65.

Simulation can be applied to investigate any system where there are limited resources that provide service, thereby resulting in delays. Examples of such systems are assembly and manufacturing lines, traffic flow (vehicles, voice and data messages, computer instructions), and customer service.

The important objects to abstract from the real system are the jobs (customers that flow through the system) and the service centers (where the jobs are processed). The primary data that needs to be extracted from the real system about the jobs is their arrival-time pattern. For each service center, we must input to the model the number of servers, the rule for selecting jobs for service, and the service-time distribution. Jobs may go through several centers before leaving the system. The conditions to send a job to a particular service center must be input. Given these specifications from the real system or from a proposed variation of the real system, the computer will then follow the jobs through the service centers, measuring such statistics as waiting time, queue length, and throughput.

The simulation program produces the flow of jobs through the network by using a random number generator to sample from distributions, a clock to advance time, and a bookkeeping code to track the individual movement of jobs and accumulate the resulting performance measures. It is important to realize that since we are using random numbers to sample from our abstraction of the real system's distribution, the output results also contain randomness. That is, the results of the simulation cannot be taken as exact, but must be considered estimates of the actual value. Typically the estimates can be made more accurate by statistical methods of designing the simulation experiment. This may involve repeating the simulation run many times using different random numbers to get several estimates, or comparing results from different portions of one run, or using longer runs.

Simulation: Example

As an example, let us assume that we want to determine the appropriate number of check-out workers in a supermarket. We look at the real system and determine that the important characteristics for the model are the arrival pattern of customers, the time they shop, the service rules and time distribution at the check-out counters, and the number of check-out counters. Further suppose that we find that the shopping time depends according to some equation on the number of items that the customer purchases, and similarly the service time depends on the number of items purchased. We then (perhaps by observing the system with a stopwatch) identify the arrival pattern by measuring the time between arrivals, estimate the shopping-time equation by measuring the shopping time and comparing it to the number of items purchased, and similarly determine the service-time equation.

Of course, not every customer arrives within a fixed number of minutes of the previous customer, nor does every customer purchase the same number of items. We would like the computer simulation, though, to use the data collected to artificially create a flow of customers that follow the actual arrival pattern and then purchase items according to the actual distribution. We therefore use the information collected to set up the pattern (or use some known distribution that closely fits the empirical data) and then sample at random from that pattern for each customer.

Suppose we capture the data and summarize it as shown in table 3.1. Furthermore, suppose that we estimate the shopping-time equation and the service-time equation to be shopping time = $5 + 2 \times$ number of items and service time = $1 + .5 \times$ number of items. This does not include waiting time on line, which will be determined for us by the simulation program.

Our assumption that all the important outcomes are known means that the sum of the probabilities is 1. If we have the computer program generate random integers between 0 and 9, the probability of a particular one of those ten integers being generated at any point is .1. Therefore, if we assign the integers 0, 1, 2, and 3 to the first item under Arrival Pattern in table 3.1 (that is, when 0, 1, 2,

Table 3.1
Arrival and Purchase Patterns

ARRIVAL PATTERN		
Interarrival Time (minutes)	Prob.	Random Number
1	.4	0,1,2,3
2	.3	4,5,6
3	.1	7
4	.1	8
5	.1	9
	<u>1.0</u>	

PURCHASE PATTERN		
Number of Items	Prob.	Random Number
5	.2	0,1
7	.1	2
10	.1	3
14	.5	4,5,6,7,8
20	.1	9
	<u>1.0</u>	

or 3 is generated, we select the result of 1 minute), the probability that the first item will be selected will be .4, as in the real system. This process is called sampling from the distribution and, over a large number of random number generations, will approximate the real system.

To see how the simulation program uses this information, let us suppose that the following two streams of random numbers are generated (stream RN1 for sampling from the arrival pattern and stream RN2 for sampling from the pattern of items purchased):

RN1: 4 7 0 3 2 8

RN2: 9 1 7 8 6 3

The simulation clock is set to 9 A.M. when the supermarket opens, and we decide to simulate the activity with two check-out counters open, one express (10 items or less) and one regular. Using the first random numbers from stream RN1 and the Arrival Pattern in table 3.1, we assign customer 1 to arrive at 9:02 (random number 4 is assigned to the second item, 2 minutes). Customer 1 shops for 20 items (random number 9 is assigned to 20 items in the Purchase Pattern in table 3.1), requiring 45 minutes (shopping-time equation) and check-out service time of 11 minutes. Customer 1 is ready to go to the regular check-out at 9:47. Waiting time is not yet known, for other customers may get in line ahead of customer 1, and therefore the time this customer leaves the supermarket is not yet known but will be determined when other customers are processed. Similarly,

following this procedure, customer 2 arrives at 9:05, shops for 5 items, and requires 15 minutes shopping time and 3.5 minutes service time at the express counter. Customer 2 is ready to go to the express check-out counter at 9:30. Customer 3 is ready to go to the regular check-out counter at 9:30, ahead of customer 1.

The simulation program thus generates the random numbers, samples from the input distributions to generate the events in the model, advances time to the next event, and then tracks the movement of the customers and accounts for the delays. The simulation experiment must be run for long clock times and/or repeated several times with different random numbers to accurately estimate the results to reduce the error in using random sampling. Getting summaries of throughput and waiting times for the two counters and then comparing these to either more counters or different service selection rules (and accounting for sampling error) will provide management with the information necessary to schedule the counter workers.

We have described some very often used computer solution methods to indicate the range of techniques, the inherent assumptions when using these techniques, and the types of applications that can be handled by decision support systems. Linear programming and decision trees provide optimal solutions for relatively static problems, whereas simulation is a descriptive technique for dynamic problems. Linear programming handles deterministic problems, whereas decision trees and simulation handle probabilistic problems. These are some of the programmable solution techniques. Decision support systems make available many operations research techniques, including different types of forecasting systems, dynamic programming, cluster analysis, and discriminant analysis, for management decision making.

The computer can support management in its need for fast, accurate, and cogent information to efficiently operate, manage, and plan the business. The combined human/computer intelligence system captures the data, stores the data for quick access, and converts the data into useful information for correct decisions. The computer provides the means to gather data, especially internal data, but also, and not necessarily for top-management decisions, to access external economic, government, and business-sector data and forecasts. Once the data is in a computer-readable form, data-base management systems provide fast access to answer queries and produce reports. Decision support systems combine mathematical solution techniques with computer computation speed and data access to assist in decision making. The technology exists to effectively use the computer to monitor day-to-day operations and support tactical and strategic planning. On the other hand, if used incorrectly, it can cause an increasing drain on capital, manpower, and time.

REFERENCES

Transaction Processing

- Kroeber, D., and Watson, H. (1987). *Computer-Based Information Systems*. 2nd ed. New York: Macmillan.
- McLeod, R., Jr. (1986). *Management Information Systems*. 3rd ed. Chicago: Science Research Associates.
- Senn, J. A. (1982). *Information Systems in Management*. 2nd ed. Belmont, Calif.: Wadsworth Publishing Company.
- Sherman, K. (1983). *Data Communications: A Users Guide*. Reston, Va.: Reston Publishing Company.

Data Base Management

- Aho, A. V., Hopcroft, J. E., and Ullman, J. D. (1983). *Data Structures and Algorithms*. Reading, Mass.: Addison-Wesley.
- Kapp, D., and Leben, J. F. (1978). *IMS Programming Techniques*. New York: Van Nostrand Reinhold Co.
- Kroenke, D. M. (1983). *Database Processing*. 2nd ed. Chicago: Science Research Associates.

Decision Support Systems

- Gordon, G. (1978). *Systems Simulation*. 2nd ed. Englewood Cliffs, N.J.: Prentice-Hall.
- Hamburg, M. (1977). *Statistical Analysis for Decision Making*. 2nd ed. New York: Harcourt Brace Jovanovich.
- Kwak, N. K. (1973). *Mathematical Programming with Business Applications*. New York: McGraw-Hill.
- Law, A. M., and Kelton, W. D. (1982). *Simulation Modeling and Analysis*. New York: McGraw-Hill.
- Loomba, N. P., and Turban, E. (1974). *Applied Programming for Management*. Holt, Rinehart and Winston.
- McMillan, C., Jr. (1975). *Mathematical Programming*. 2nd ed. New York: John Wiley and Sons.