الجمهورية العربية المتحدة

التخطيط القومي

معهد التخطيط القومي

Memo. No. 568

INFORMATION SYSTEMS FOR PLANNING AND CONTROL

By

LIONELLO A. LOMBARDI

OPERATIONS RESEARCH CENTER

May 1965

القاهرة

[illegible]

Information Systems for Planning and Control

Summary

The current state of management information systems is discussed. Information processes are classified in two categories: "self-contained" and "data base" and it is pointed out that although the ones belonging to the first category are by and large less important to management, they are at present the only successful ones. New methods of designing data base processes are proposed. The concept of "real time" in put into perspective and the "timing paradox" is brought up. New methods of implementing data base processes are discussed: they follow the classical division in "collinear" and "on line". The "maximum parallelism principle" is presented. It is shown that the most efficient collinear implementations of processes no matter how complex should utilize no more than four tape units ("four tape principle") or two disk-pack units ("two risk principle").

A new type of non-time-sequenced, policy-oriented programming language[1] and its complement in terms of a software aid (the "interfacing" program) are proposed as tools for providing management with flexible means of accessing diversified information and implementing policies. By means of this language, policies (which are designed without constraints and expressed without referring to their implementation) are made totally independent of the way they are carried out by the information system. Hence the same language (with different is implementations) can be used regardless of whether the implementation is collinear, on line or hybrid.

[1] A description of this language, named N1, is contained in 16 and hence is not repeated here. But, except at most for the second half of section 4.4, the contents of this paper can be grasped without a technical understanding of N1, i.e., without previous reading of 16.

PART I. A NEW CLASSIFICATION FOR INFORMATION SYSTEMS

1. Introduction

1.1 Much expectation and little accomplishment

The first computers were built in the 1940's. They were called electronic brains, with reference to their potential to take over typically human thinking functions. But while they have turned out immediately to be useful in performing huge arithmetic computations needed by engineers and scientists, their progress towards "thinking" has been much slower.

When in the early 1950's computer(and, more in general, information technology) were first made available for commercial or administrative purposes, it soon became common to think that they would have a direct impact on management both by taking over some managerial functions and by facilitating others by supplying managers with typically managerial information. So many started speculating that executive functions in business, government and military areas would become increasingly dependent upon computers as tools to provide an information input for decisions. That in order to use computers effectively, top executives should learn to understand a wide spectrum of information system concepts in areas such as command and control, financial planning, comparative analysis of alternate courses of action, information retrieval, etc. That they should develop competence in the use of automatic information systems in management planning and control. That the executives' organizational training, decision-making experience and economical motivation would envolve into factual knowledge and critical understanding of the information and man-computer environment where many mathematical techniques can operate sucessfully. That, although not being able to solve every information-technological problem that he encounters, the executive would eventually be in a position to determine applicable techniques, select appropriate equipment, identify structural inadequacies, evaluate the contribution of specialists, identify problem areas where expert help is required, and recognize a difficult problem when he is confronted with one. That finally the top executive would develop the same confidence in using quantitative principles for making decisions on the basis of information provided by integrated, semi-automatic, computer supported information system that he now has in making decisions in the conventional way.

Management analysis and sociologists have devoted significant research to prognosticating the possible changes of the structure of business organizations and of the whole of society that this would yield. These studies were all based on the assumption that computers actually do have the potential for either replacing managers or to work <u>coextensively</u> with them. For example, the work of Leavitt and Whister 1 shows that information technology has in herent elements which could create a pressure to recentralize certain parts of organizational structures. On the other hand (but not necessarily in contradiction) Burlingame 2 showed that other aspects of information technology may create the possibility of decentralized, strongly particupative, loci for some classes of management decisions. And the work on Anshen 3 reveals several factors which set an upper bound to the rate at which acceptance of information technology can take place.

But when one reads these or other works, all based on the same assumption, and at the same time looks at what is going on in companies which have introduced computers, a question arises in the back of his mind: are computers being or heading to be used in typically managerial functions, or are they or will they in foreseeable future be just handling clerical routines? Is there a trend of information technology to interact with decision-making, or are computers relegated to accounting? Can information technology directly or individually help in the few, relatively few, but important, decisions made by executives, or just in the numerous but secondary ones made by clerks?

Dearden 4 phrased the question, and gave the answer. In his courageous and realistic article he showed that there is a wide and widening gap between dreams and reality. That the areas of business administration on which information technology has in impact tends to be mainly routine, hardly qualifiable as management functions, while information technology has little effect on upper echelons of management. And, most important of all, that the very existence of any substantial trend to rapidly change this situation is doubtful.

In industrialized countries the usage of computers is expanding at a rate much higher than that one of their national economies. But the room for expansion is provided chiefly by clerical paperwork, as opposed to managerial functions. In this sense, the change which is taking place as a consequence of the availability of computers is similar to the one caused by automated production, which affected drastically the blue collar worker, but not directly the design engineer. Of course some managerial functions are occasionally being taken over by computers, but the abundance of literature covering each of such events is a symptom of their sporadicity of occurrence.

There is no incumbent revolution due to computers in the technology of management. And if there is an evolution, it is taking place by and large more slowly than the current evolutions in many other technologies.

Now why is it so? The reason, we think, is that the information problems of management have not yet been studied in terms of automation. There are good computers available, there is a considerable amount of knowledge about decision processes, organizational structures, and optimization/oriented mathematics.- But the problem of structuring the role of automatic information systems within an organization in a way to make it inherently amenable to play a role in management has not been studied yet. There is no body of knowledge about this area, that we can call <u>management information systems</u>

analysis, not to speak of a repertoire of techniques. (Not being a sociologist, this author has to register this situation as a fact, without being able to explain it historically). This situation implies that the picture given by Dearden will continue to represent reality until basic theoretical guidelines and an appropriate technology are developed. This requires a major leap forward in theory to fill the gap. When this will be achieved, but not before, the picture might change.

So this article is devoted to presenting some initial results of studies of the theoretical structure of management information systems and ensuing technological guidelines to their development. At least, this paper proves that, although it has never been done befor in any real sense, such a system can be developed, but only conditional upon adopting several entirely new concepts and methods, and departing from conventional approaches.

The topics discussed here are:

- The correct relations in terms of information flow between operations research, information systems analysis, management, and data base.
- A model of information system which is immediately responsive to management's requests for information and to order of implementing management's policies.

Such a model does not impose any constrint on information requests or policies.

- The response time requirements of different classes of information requests and policies, and their reflections on the way of implementing the information system.
- A method of implementing information systems based on that model with a minimum amount of hardware.
- A discussion of a programming language and other software aids to management information, also based on the model.

. A method of planning the implementation of management information systems of any level of complexity at minimum cost.

1.2. A state of confusion

When approaching the problem of evaluating the contribution that computers and operations research (i.e., information technology) can give to his organization, or the one of selecting appropriate data processing equipment, or of identifying specialized techniques which are applicable to particular information problems, an executive today is confronted with a large amount of partly unstructured, incomplete, often contradictory criteria and methods, of which he sometimes can hardly make sense. He hears that computers can be very fast, accurate, efficient and economical in performing routine computations and report preparation; that the formulation and computer implementation of models of the organization or parts thereof can sometimes help in evaluating the implications of proposed policies, sometimes issue optimized operating decisions, while sometimes they cannot be of any substantial help at all; that there are various techniques of building models and simulations, which apply to different situations; that there are various "artificial languages" which facilitate and speed up the programming of the work of computers in some cases, while in other cases the programming work represents a major investment, and changes of a program involve major expenses, difficulties and a long lead times; that areas of mathematics called mathematical programming and statistical analysis can bring to light important facts only conditional upon the raw information available meeting certain requirements; that different information requirements need different configurations of equipment. But what he does not find is some general criterion which enables him to see what fits where, which techniques apply to which problems under which conditions, which ones of the techniques that he needs are available today, which ones are not available but can be developed, and which ones depend on unsolved problems, still in the domain of basic research.

The first aim of this report is to provide the executive interested in planning or evaluating management information systems with some orientation in this maze, to make him aware of the different stages of development of different tools, and especially to what situations and under what conditions certain tools apply.

The second aim is to present the results of new research on the information flow in an organization, which leads to a flexible model - the "functional polyhedron" - which is common to many organizational structures in various areas. This model yields new ways of designing information systems, of selecting hardware to support them, and of building languages to program them.

1.3 Identification of problem areas

As we said, when computers first became generally available in the early 1950's, much emphasis has been given to their potential in extending the human intellect and thus helping at high level of decision-making. However, in current practice, in most organizations computers are confined to a low echelon role: role consisting mainly of routine accounting and routine engineering computations.

Often one thinks that this is only a starting point, and that the next step is to gradually extend their role to increasingly high levels of decision making by rationalizing and reducing to detailed sequences of instructions the decision procedures involved. But when this work is faced, one is confronted with the problem that many decisions require the simultaneous presence of heterogeneous information, whose supply must be taken care of when the computer programs of the information system are written. Hence, those programs must handle the simultaneous flow of different streams of data. Furthermore, changes of policies may involve changes of the information requirements, and consequently re-writing of such programs. This is a major job which often discourages initiatives; in fact, artificial languages now available to help in this work ("autocodes," COBOL, IBM Commercial Translator, FACT, etc.) are useful only in freeing the programmer from references to the detailed structure of machine instructions or frequently used sequences of instructions, while they involve no major tool for specifying the control pattern of the data flow in conciese form.

Before and during the computer era, mathematical techniques for data analysis have been developed extensively. Statistical analysis, mathematical (i.e., linear, quadratic, integer, dynamic, etc.) programming, and network flow analysis (i.e., PERT, Critical Path Method, etc.) are well known and documented technological areas. But the limitation common to all mathematical manipulations is that they require correct, meaningful and accurate data. The output of sophisticated mathematical manipulations is correct, meaningful and accurate only to the extent to which the raw data were such. Meaningfulness of raw data in terms of exact knowledge of what exactly each number is a measure of and of where and under which conditions such measure was taken, is often the most essential (and often overlooked) factor. Now such raw data are often provided by sections of information systems other than the one which analyzes them so that the usefulness of the final results depends on these. Mathematical manipulation, far from being a weak point, is at present by far the strongest link of the chain of phases of the metabolism of information in organizations. Yet it is a link in a chain, not an alternative approach which can replace other, weaker links. Since the total strength of a chain is the one of its weakest link, at present most attention should be concentrated on other phases of information processing. More precisely, one needs methods of providing the exact type of information which is needed for the particular mathematical analysis that one wants to adopt, and methods of monitoring the meaning of data in terms of their source and the path through which they flow.

Decisions tend to become less and less routine the higher one goes in the organizational hierarchy. By reflection, the information needs for functions performed at higher echelons are quite variable. This applies both to the case of automatic and human decisions. So an information system can effectively serve at this level only to the extent to which it is possible and easy to substantially modify its output of information at various points, in order to issue information that nobody had requested before, which a portioriis of a nature that had not been contemplated when the system was first built. This can be achieved only if the general structure of the information system is independent of the mode of operation of the information inlets and outlets, in the sense that each time one decides to add a new inflow of information or to retrieve a particular type of information, one should not necessiatate redesigning the whole system, but simply to place an inlet or outlet at an appropriate point, or modify existing ones. In particular, such outlets should consist of the implementation of a formal definition or representation (<u>filter</u>) of the information which is needed. This requires both a systematic way of designing and controlling the data flow and an artificial language in which to write the above representation.

These three problem areas - programming complex integrated systems, monitoring the meaning and value of information, and augmenting systems with information inlets and outlets [illegible] at unanticipated points of call for advacement in two major areas: the analysis of the flow of data and the design of appropriate system languages. These two areas are intimately doubly related, first, because the data flow can be analyzed, controlled and modified only if one has an appropriate language to do so; and second, because a language to design inlets and outlets, which should be independent on the data pattern, can be conceived only on the basis of certain assumptions on the data flow which are valid in general.

The following section of this report will focus mainly on these two areas.

1.4 Relation between operations research and information systems analysis

A typical mistake in talent allocation accounts for much dissatisfaction about management information systems. This mistake consists of intertwining mathematical techniques and data flow organization. Since the first are currently at a higher stage of development, the second has often been viewed as playing a purely accessory role, and designed in unrelated bits and pieces, each serving some particular contingent need of the first. The result has often been that the whole system was unsatisfactory, although the mathematical parts might have been correct. This mistake has often been caused by a misunderstanding about the role of operations research in an information system. More precisely, a typical pattern of this mistake consists of a tendency of systems analysts with a background limited to operations research to focusing their interest on the decision-making process, and may-be trying to improve it mathematically, disregarding the delicate function of supplying information for such process.

The latter function can be accomplished only in terms of methodic design and analysis of the information context in which decision processes take place. To design controls over such environment, which allow it to provide information input for the decision-making process, is the primary role of the information system specialist. To analyze and improve the decision process is the role of the mathematician or operations researcher (whose tools, however, we will not discuss in this report). So it is very important to keep apart these two roles, which involve different and unrelated techniques.

Decision processes are sometimes delicate and difficult to analyze, be-cause they generally involve intangible factors which it is hard to rationalize. For example, the choice of variables in the process or the one of the group of persons who performs it can depend on historical and organizational peculiarities, not in line with organization theory. Sometimes such imponderable factors play a very useful role in the decision process, and often trying to modify it in the name of mathematics may involve shocks of unpredictable consequences on the organization. The rapport between decision process (with or without mathematical help) and information system is that the second furnishes information input to the first. But an information system should not be allowed to contain as integral and unseparable parts decision processes, although these can be added as separate and removable modules. And it is not the business of the management information system specialist to get involved in evaluating the decision process. Rather, this specialist should design the information system in a way such that any information can be fed in or retrieved wherever and by whomever wants to do so, without concern as to whether or not particular inputs or outputs are desirable from a managerial standpoint: he should enable the manager to decide about that and modify his designs when he wants, without in turn requiring that this one be concerned about the structure of the information system.

This separation between information system and managerial and/or quantitative decision process is fundamental in the concepts discussed here. One of the purposes of the work reported here is to try to provide tools by means of which the information system analyst can perform a useful task in improving the output of management decisions without getting involved in delicate organizational matters. Our brightest hope is to help analysts to reduce somewhat the friction and mistrust which so severely hamper their contribution.

The job of the information system analyst is to provide economical and timely information upon the specifications given by management. As of today he has not yet any effective equipment of notions and techniques to perform this job. The following section of this paper report on work aimed at starting to build such tools. On the contrary, the kind of information which is drawn, (e.g., exception reporting) the places where it is made available (e.g., who should receive which reports), and the way such information is utilized are matters of policy, not matters of system analysis. The golden rule of discipline of the information system analyst should be "stay away from policy matters." Instead, his duties are to make any type of selection, point of outlet, and utilization of information possible, economical and timely.

The lack of insight and of a body of knowledge about management information systems yielded incompatibility between requirements of stabilityn of operation vs. flexibility and change. Hence there is now a dichotomy between ongoing, low echelon, routing processes and advanced experiments. One of the goals of the methods reported here is to overcome such dichotomy.

2. Data bases and their evolution

2.1 Self contained processes and data bases

Some kinds of information processes have been more successful than others. However, there has been so far no rationale available to determine underlying causes of success or failure, so we can not utilize past experience for prediction. We think that there is a measurable reason for difference in results. Here we will propose a simple taxonomy of information processes based on two main categories: <u>Self contained</u> processes, for which tools are already widely implemented and documented, and <u>data base</u> processes, for which a theory and technology, not yet existent, is proposed in this report. Most of the successful processes of the past and present belong to the first category, while the second is much more important in scope, capital investment, and impact on organizations.

Let us identify self contained information processes: such is a process which, each time it takes place, is preceded by a reading of <u>all</u> of the information input that it needs. Such reading (which may refer to gauges, prices of the security or commodity market, levels of inventory, backlog of orders, etc.) will be called <u>input experiment</u>.

Typical examples are: optimization by linear programming of oil, feeds of chemical blending processes; regulatory controls of industrial processes; automatic numerical programming of metal cutting machines; most current job shop simulations.

If (and only if) the input experiment can be handled sucessfully, the only remaining difficulty stays in the internal processing of the information. If effective tools for this purpose exist and are utilized, success is bound to come. As a typical example of success we may quote the linear programming optimization of the operation of oil refineries. As example of a case where the input experiment can be handled properly, while knowledge of the internal processing is less complete, we may quote industrial process control or control of spacercaft, where, in fact, success varies reflect-ing variations of the extent to which one knows how to formalize the decision structure of the control programs for particular purposes. As example of a case where effective internal processing techniques are available, but the input experiment often presents difficulty (and, accordingly infers, variations into the total effectiveness) we may quote PERT project scheduling systems.

Most computations stemming from research and development in engineering and in the physical, life, medical, social, or earth sciences, are typically self contained. In such processes the input experiment consists of reading requirement specs, lifting number from the literature, or carrying out measurements or experiments (in the conventional meaning of the latter word).

So the bottleneck is internal processing, chiefly of mathematical nature. And the fact that there is today a whide body of knowledge in mathematics explains the fact that, in many cases, information processes in these areas are successful and helpful.

An interesting feature concerning many self contained processes is that the input experiment involves little information. Consequently there is little interfact between the information process and the organizational environment, and hence little interaction between the introduction or modifications of the first and the structure of the second. This situation is responsible for facilitating adoption of self contained information processing methods in many organizations.

The second class of processes, namely <u>data base processes</u>, is identified by the fact that, in addition to data resulting form input experiments, each time a process takes place it draws data from and/or modifies an existing data base. The significant cases are those where such data base has the following properties:

- Is permanently in existence, and so establishes continuity between repeated performances of the same or different processes.
- Is diversified and structured, as opposed of consisting of arrays of numbers having similar hierarchycal level and analogous connotations.
- Is common to all information processes within an organization, and hence covers a whole organization, and perhaps is integrated with the data bases of other organizations, as opposed to being associated to one particular process.
- Each process, each time it takes place, uses as input and modifies by its output only a selected, small part of the data base. However, the particular part which comes into play varies as a function of the process and of the different execution of each process in a way that it is not completely predictable in advance.

Is evolutionary, in the sense that it keeps changing both in structure and contents. In other words, each time a process takes place, it will find the data base modifed by other processes (or by a previous execution of the same.)

A data base is the body of standing formalized and machineable information available within the organization. The extent to which the data base covers all information relevant to management decisions (i.e., extent of automation of the management structure) varies from one organization to the other, and, within the same one, it varies and should increase with growth and maturity of the organization. Ideally, in a highly automatizied organization, management decisions (with or without mathematical help) should be based chiefly on information drawn from the data base (as opposed to the physical environment of management or listings periodically printed in hard copy); the results of such decisions should have as first effect modifications on the data base and they should affect the environment only indirectly.

The information system is the part of the organization which provides for:

- Organizing and marshalling the data base
- Controlling the transfer of information between management and the data base and vice versa. In order to do this, the information system should have very flexible programming tools (not yet available today, but outlined in this report) allowing immediate implementation of selection criteria, freely specified by management, for extracting information from the data base on entering information into it
- Controlling the transfer of information between the physical environment (plants, outside correspondents, etc.) and the data base.

The information system consists of computer hardware, a package of program generators (which will be discussed later) and a staff of information system analysts. To the extent to which the criteria set forth here are followed, no application programmers (in the conventional sense) should be necessary, although need for developers and maintainers of the programming tools is to be contemplated.

A data base is broken down into files, each consisting of a set of records. A record is a complete characterization of a given item. for instance, a record may contain all information pertaining to a given item in inventory, or to a customer account, or to an employee, or to a taxpayer, or to a design project (in which case it may contain blueprints and/or references to blueprints). Or it can also contain the company correspondence with a given correspondent, or summary information concerning a technical paper, a list of titles of technical papers on a given subject, an abstract of a law, all court decisions on a given issue, complete information on a patient in a hospital, all information available on a specific disease or on a sector of the national or local economy, population, type of employment, etc. A record can also be a program for quantitative (statistical, etc.) analysis. A record is broken down into fields, each of which can be in turn broken further down into other fields, and so on. (Some fields may contain information directly amenable to graphical representation by means of visual displays.)

Each record within a given file must be unambiguously identified by the contents of one or several of its fields, named key. Typical keys are the stock number of an inventory file, the social security number of an employee or hospital patient, the project identifier for a set of blueprints, etc. These meanings are called key connotation, and identify a file. A file does not necessarily have to be sorted with respect to its key, although this can always be done. In a properly designed data base there should never be more than one file with given key connotation.

Items of information flowing between the data base and the environment, between the data base and management, or between two different files ~~of~~ the data base are called messages. In principle, message has a structure similar to the one of a record, although in most cases it is shorter. A message orginated within the data base by the operation of the information system is called _endogenous_. A message coming from management or the environment is called _exogenous_. A message is called _endodirected_ or _exodirected_, respectively, depending on whether it is directed towards the data base or the outside (management or environment).

No message handled by the information system can be both exogenous and exodirected. A message which is both endogenous and endodirected is called a _transfer_ message, and it never leaves the _information system:_ hence the existence of transfer messages if of concern only to the system analysis.

Exogenous messages coming from management may be requests of displays of single records or sequences of records satisfying a given conditions, requests of compact summaries (e.g., totals, counts, averages, or results of mathematical analyses based on fields of all records of a given file which satisfy a given condition). They can also be expressions of decisions, such as orders to buy, sell or produce, to change fields of specific records or of records satisfying given conditions, or delete or add new records to files.

Exogenous messages coming from the environment can be keypunched and formatted summaries of letters (e.g., orders), readings from gauges in plants, coded wire messages, etc.

Exodirected messages can be in the form of letters, printouts, displays, wire messages, pulses to plant regulators, etc.

Examples of data base information processes are all those which pertain to a diversified operation, such as integrated management of continuous or job shop production, raw materials, parts and product inventory, supplies, personnel, finance, sales and shipments; command and control of entire weapon systems and their supports (as opposed to command and control of a single spacecraft, which self-contained) or of diversified military forces; reporting and cross-information systems for project management, involving time schedules, expense accounting, keeping track of work force, and retrieval of blueprints and memos.

Data base information systems are of interest to the management of complex business, civil service or military organization, while self-contained information systems are typically of interest in specialized technical areas. Data base systems are the most important and so far less studied. In this work we are mainly concerned with data base systems.

A hardware consideration: much has been said about the fact that the decreasing cost of computing hardware will eventually make small, powerful, future computers so inexpensive to make them easily available in every office. Accordingly, in the controversy about the relative convenience of separate small computers vs. multi-access, large ones, it appears that the first ones will eventually have the upper hand. This might be true for usage related to self-contained information processes, but not to data base ones, because compartmentalization of hardware would prevent the usage of a common data base to different processes within an organization. We feel that, in the perspective of the predictable advances in electronic engineering, this one, as opposed to scale economy and saturation of capacity, is the strongest argument in favor of multi-access approaches.

2.2. Mathematical techniques require self-contained information environments.

If we look at past experience with operations research, we note a remarkably positive correlation between success and self-containedness of the applications. So the first question one should ask when trying to apply established mathematical programming or probabilistic analyses is: Is the information process that we are considering self-contained? In case the answer is negative, this does not mean that OR is useless. Instead, it means that a prerequisite to the OR study is a study and possible modification of the information system aimed at providing controlled outlets of unambiguously identified information which serves as input for the OR process: if this cannot be done successfully, then OR efforts would be misplaced.

To do this successfully, one should try to adjust outlets of information in a way such that the OR process takes place in its habitat, i.e., in a selfcontained environment, characterized by the possibility of meaningful input experiments. When pertinent information is contained in an on-going data base, such input experiments should draw directly from the data base, by means of an interface, and directly from the physical environment. Such interface can be designed in terms of judicious placement of outlets in the various processes on the data base. So, by building an interface appropriately, one can create locally a self-contained situation within a context of data base processes.

In current practice we occasionally see this approach implemented. For example, some inventory control programs (which are typical, even though simple, data base processes) for each inventory item one sales forecasting and inventory optimization subroutines (which typically perform a self-contained function) and supply it with up-to-date pertinent pieces of information selected and drawn from the data base (chiefly the inventory file, whose records carry information on past sales).

Actual implementation of such phylosophies with present tools is hard, expensive, and hence involves rigidity with respect to the evolution of policies: consequently we rarely see them working, and almost never see advances beyond the point epitomized by the above inventory example. However, the techniques suggested in the last part of this report can drastically reduce implementation efforts.

A similar positive correlation between success and self-containedness is evident in simulation studies. The main argument against the significance of such studies does not affect the simulation techniques themselves, but rather the fact that they can be implemented easily only in fictitious, artificial, hard-to-test environments. In fact, by generating an artificial data context one easily meets the prerequisite of the self-contained situation: but in this way he also loses contact with the real context with which one is interested in observing simulation models interact. Such context can only be supplied by the data base of the real organization, in terms of appropriately placed outlets. So the future of simulation depends on the extent to which we will be able to create local self-contained environments for models within the framework of data base.

A third area of problems worth mentioning is the one of project management. For this purpose there is a whole selection of effective mathematical techniques going from elementary algorithms (such as PERT and critical path methods) to quite sophisticated network flow analyses. But all of such techniques operate well only in a self-contained environment. This explains why most of the difficulties in implementing them stems from supplying up-to-date and significant raw data to the algorithms. Such data must come from a data base, which, is project management, tends to be even more evolutionary than in the management of continuous operations. Hence, what is needed in order to solve the difficulties arising in this area is again to enable us to perform input experiments on the data base, which is independently maintained by various data base processes. This again can be achieved only in terms of techniques of plugging modular information outlets in arbitrary positions in all data base processes.

2.3 Timely information supply and the timing paradox

The expression "real time information process" has become very popular, although not everybody agrees on its meaning. We will adopt the definition of E.L. Glaser: "An information process takes place in real time if, upon the receipt of outside (input) messages, it initiates, stops or modifies its output in time for taking appropriate action." By this definition, an information process is good only if it takes place in real time. The characteristic parameter describing a system from this standpoint is the mean lead time or response time (measured in units of time) between the receipt of the input and modification of the output.

The adequacy of the response time of information process should be evaluated with respect to its relation to the time delays implied by the transmission of the input messages and the execution of the actions provoked by the output. For example, in case of a process which controls the shipments by trucks of inventory goods to other cities (which takes a time of the order of days), a response time of hours or up to one day is quite tolerable for considering the process as a real time one; the same holds for processes involving the control of payments made by customers, since it takes days to forward and deliver the mail. In case of computer scheduling of a job shop where holding times in the different machines average hours, the response should not exceed a few minutes. For the control of a missile of an airplane in the environment of an airport, highway, it should not exceed fractions of a second. At the other end of the spectrum, a process which decides on which taxpayers further checks have to be performed is real time even if it has a response time of weeks or months, because of the time necessary to execute such checks.

A master technique of reducing the response time of a process will be discussed later. In general, however, expediting the response involves a certain marginal increase in capital investment in equipment. So the marginal profitability of efforts aimed at collapsing responses below the level at which processes take place in real time would be negative. So we need quantitative guidelines to evaluating the response time of a system.

Let us consider as example an information system designed to handle the complex scheduling and execution of a large scale military move which takes several days to execute. When such a system is asked to actually perform a move, a response time of hours would probably be adequate. But the very same system should be able to respond to another kind of external orders, namely requests for information about all the various implications of proposed moves, on the basis of which such moves can be evaluated. As a consequence of the information issued by the system, a proposed move may be discarded and information on an alternate move asked. Here the action initiated by the output is not an actual move which take several days, but rather a staff discussion which takes a few hours or minutes. So the response time must be considerably shorter (of the order a minute at most).

Similar examples can be found in business. For instance, the response time to a message entailing the order of raw materials, that it would take days to load and ship, can be of several hours. On the other hand, a request by a purchasing executive who wants information about prices or implications of several different proposed purchasing policies before he makes an operating decision, needs to be answered in seconds or minutes.

In the control of an industrial manufacturing process whose inherent response time to charges of regulation parameter is of hours, an information system with a response time of tens of minutes is adequate for performing actual control. But a shorter response time is needed when the information system has to rapidly predict (on the basis of a built-in model of the process) the consequences of a change of parameters that a process engineer would like to test in order to decide whether to make or discard them.

As last example, consider airline reservation systems. Here a response time of tens of minutes is adequate to actually place a seat reservation. But requests of seat availability information by a customer who is considering alternate travel plans must be answered within the time of a telephone call.

All these examples are drawn from data base processes. All of them show that there are two different kinds of input ~~mess~~ messages in an information process which require two substantially different response times. So the problem comes up of identifying the two types of input messages and utilizing such distinction in system design.

The situation depicted in those examples is quite frequent in data base processes. Formalized, it sounds paradoxical: Input messages into a data base process require a much faster response time when they do not directly imply the execution of any physical action, i.e., when they are dry information messages. This phenomenon is called the timing paradox of data base information processes.

The timing paradox is characteristic of data base processes. As counterexample, in air traffic control in the vicinity of an airport (a typical self-contained process) responses of fractions of a second are much more necessary when their aim is to effectively adjust the behavior of airplanes in flight than when their purpose is to answer to dry information messages.

The timing paradox is not a universal truth, but a quite informative standpoint for initial drafting of data base information systems. It should be kept into account to avoid heavy unbalances. Its utilization works as follows: if a process has to ~~a~~ accommodate dry information messages, then the response time should be of the order of the time necessary to apprehend the information issued as a result. If not, the response time should be not larger than a fraction of the predicted time needed for executing actions based on the information output.

The need for accommodating dry information messages often involves higher marginal capital investments which do not reflect in a more effective execution of action. The justification for such extra cost should be found in terms of a wider and more selective supply of information to management and of the fact that many important executive decisions have several dry information messages as a prerequisite.

3. Collinear Data Base Information Systems

3.1 General concepts and the maximum parallelism principle

Collinear data basis information systems are data base systems which processes the data base in a way which satisfies the requirement that

Any time a file is called into play, all messages available in the system, which are directed to any one of its records(called input messages), are completely processed. Such complete processing of all messages directed to a single file is called a function. Within a function records of the file involved are brought in sequentially, and each one is processed until exhaustion of all available messages directed to it. The complete processing of a single record is called a phase. So a function is executed as a sequence of phases. The time elapsing between successive executions of a given function, say, fi , is called period of f_i.

A data base information system should involve as many functions as there are files. The execution of a function involves.

- Reading all input messages and executing all action that they call for
- Replacing the old file by means of a new (updated) file. The old file is no longer of any use (except for emergency reruns in case of machine breakdown) after the end of the

execution of a function

- Prepare and issue all endogenous (exodirected or transfer) messages (output messages) implied by the interaction between the file and the batch of input messages.

By and large the time necessary for executing a function on a computer is only by a little fraction greater than the time necessary for reading the old file and writing the updated one. This is the substantive reason for the desirability of handling at once all input messages, no matter how diverse their natures and origins are, and hence issuing at once all kinds of different output messages. This is called the maximum parallelism principle, and the parallelism of a function can be defined as the total number of types (streams) of output messages that it issues. Applications of the maximum parallelism principle is desirable because it allows

- Efficient use of capital for equipment, or alternatively for enabling to handle a given data base system with less expensive equipment
- Avoiding time wasting repetitions of the execution of a given function. Hence being able, at the same cost, to have a more frequent execution of each of the functions. This can greatly accelerate the response time of the system to exogenous messages.
- Making all programmed decisions with a wide synopsis of all information from all possible sources, including the items of information which came at the last minute.
- Keeping the data base as up-to-date as it is conceivable, thus allowing it to "live" and to be a reliable source of information
- Augmenting and modifying the scope of the information system (i.e., adding new streams of messages, hence allowing

management to obtain new kinds of reports or answers to unanticipated types of questions, etc.) at negligible marginal cost, because such extensions do not require additional scanning of files.

However, most systems in existence today have a very low parallelism, (i.e., they involve repetitive scanning of files for processing different streams of messages). For example, a frequent and extremely harmful practice is to separate the updating of files from all other operations. There are reasons to believe that the cost and response time of many present day information systems could be reduced by a factor of ten just by redesigning them at maximum parallelism. But what is the motivation for such an undesirable practice? Or, better, what is the difficulty in adopting sound ones?

The difficulty is that programming a highly parallel function generally involves writing a very large and complex program. To originate such a large program and to adapt it to the changing requests of management is very costly and slow.

Unfortunately there are no effective tools to help here. COBOL and similar languages do not facilitate this work: all uses we know where such languages have given some satisfaction were functions with extremely low parallelism. So the very possibility of adoption on a large scale of the maximum parralelism principle is contingent upon the development of the substantially new type of programming language that we advocate.

3.2 Routing of transfer messages

Collinear data processing involves cumulating all exogenous messages entering the information system. More precisely, it involves grouping them in different batches (input batches), each one corresponding to a file. Each such batch is used as input, and thus annihilated, every time the corresponding function is executed.

This implies that the first thing to do on an exogenous message is to assign it to a particular batch. Then, as part of the next execution the function (let us devote this function fi) the interaction between this message and the file may issue exodirected as well as transfer messages. In particular, the program of the fi must take care of encoding into each transfer message some indication (code) of the function to which it is directed. At the end of the execution of fi , each transfer message issued by fi will be batched together with all exogenous and other transfer messages (coming from any function) to provide an input for the particular function to which it is directed. This relaying of messages (i.e., exogenous - transfer - ... - transfer - exodirected) is called internal routing.

For example, when an exogenous order enters the information system of a wholesale company, it should be first batched with the input for the inventory function (whose file, the inventory file, has the stock number as key connotation). The impact of this order upon the corresponding record (i.e., the stock record of the item ordered) cannot produce directly a shipment order, because at this time all there is available is information on the item ordered and none on the party who issued the order. What is generally done instead is subtracting from the quantity-on-stock of the record the quantity ordered and issuing a transfer message to the function corresponding to the customer file. This transfer will contain the code of the customer as well as all information relevant to the shipment which has been gathered as a result of the impact of the order message with the inventory record.

The customer's function may have as input exogenous messages such as payment slips for accounts receivable, transfer as the above for clearing shipments and issuing invoices, or requests of specific information on specific customers, changes on the customer file, etc. As required by the parallelism principle, each time it is executed it performs at once all the processing of the customer file, i.e., account receivable, updating of the file, invoicing, information retrieval from this file, etc.

Let us assume that the impact of the above transfer upon the corresponding customer record causes the issuance of an order-of-shipment and other exodirected messages (e.g., letters, bills, etc.). If the company's invoicing policies encoded in the program for the customer's function require delaying the issuance of the bill, then such impact will issue an appropriate transfer directed to the next execution of the very same customer function (the only function which can handle bills) for later invoicing. Otherwise a bill is issued as direct result of such impact.

If, on the contrary, the information on the customer is such that, according to the company's policies, the order should not be fulfilled, then, besides perhaps appropriate exodirected messages (a letter to the disappointed customer, a report to management, etc.) the function should issue a (feed-back) transfer directed to the (next execution of) the inventory file containing information sufficient to add back to the amount-on-stock of the item ordered the amount of the shipment that has been cancelled.

In case the stock item considered should be reordered according to the company's policies encoded in the program for the inventory function, then the first mentioned execution of this function should issue a tranfer directed to the suppliers' function. This transfer will contain the code of chosen supplier (the list of all suppliers of the stock item considered must be contained in its inventory record) as well as all information relevant to re-ordering (quantity, urgency of shipment, packaging instructions, etc.), which can be lifted from the inventory record. Then, under normal circumstances, an exodirected order will be issued by the next execution of the suppliers' function (which, for parallelism reason, will also handle ordering, accounts payable, updating of the suppliers' file, reporting from such file, etc.).

In the above example, routing could be handled differently, i.e., the exogenous order could be first given as input to the customers" function, which in turn would issue a transfer to the inventory function. Internal routing is an important decision to be made by system analysts because it may influence the response time to messages. In general, the systems analyst can regulate the response of the system to messages by operating on the following sets of parameters:

- Periods of the different functions. Sometimes a simple round-robin is adequate, while in many cases some functions need a much shorter period (and, consequently, often handle smaller input batches) than others. Variable periods for a given function, as well as unscheduled executions aimed at handling urgent messages, should also be envisaged.
- Routing of the different streams of exogenous and transfer messages.
- Matching the machine power to the load implied by the periods of the functions.

The above are the points which the work of the system analyst should focus. This "dynamic" analysis approach represents a major departure from conventional systems analysis, which is limited to "static" considerations about the average frequency of occurrence of messages. Hence conventional systems analysis does not give insight into the dynamic of an information system, and does not help in improving responses nor in reducing the requirements on machine power. (For conventional system analysis see the IBM manuals on SOP-Study Organization plan, or, for excellent presentations, reference 5 and 6

3.3. Local self-contained environments

Referrring to the above example of the information system of a whole sale company, it occurs that decision concerning reordering, which are part of the inventory function, can be made rationally and optimally made with the help of modern mathematical inventory management techniques, mainly based on time series analysis (such techniques are widely covered in the literature). As it is clear from section 1, however, such techniques apply to self-contained systems, while the system presented in the example is typically a data base one. Is this a contradiction, or can we still apply such statistical techniques?

We can: more precisely, we can apply them on data that the inventory function provides, thus creating a local self-contained information environment with respect to each inventory record, i.e., each phase. In order to do so, mathematical reordering policies must be encoded in the program of the function, and the data relevant to them (records of past shipments, inventory cost figures, etc.) must be contained (and updated by each execution of the inventory function) as fields of each inventory record. Inventory records can also contain parameters for the programmed policy in order to allow the program of the function to choose between alternate re-ordering policies which apply to different stock items (the design of such parameters is a management, not a systems analysis decision). In advanced applications, one can think of pushing diversification of policies from one stock item to another to the point of encoding a part (subroutine) of the reorder policy program in fields of inventory records, thus sharing the policy program between the program of the inventory function and its data (in this case, the inventory file). In any case, the mathematical reorder subroutine is executed for each phase during (as opposed to after) the execution of the inventory function.

More in general, self-contained environments to which operations research can be applied can often be created <u>locally</u> for each single phase, during the execution of a function. In this case assistance of operations research specialists is required in order to

- Design and program the mathematical algorithms involved
- Tell to the system analysis which information should be placed in fields of records, so that such algorithms can operate by finding their habitat in terms of a self-contained environment.

On the other hand, the system analysts should be charge of

- Making the above information available in the files
- Designing and programming the functions (all of them, not only the one directly involved) so that such information is kept up-to-date Incorporating the programs written by the operations research people in the correct position of the program of the function directly affected

Proper allocation of tasks tothese two different technological specialties and interaction between them is vital for success.

3.4. <u>"Summary" self-contained environments</u>

Local self-contained information environments within functions is one of the two ways operations research and, in general, mathematical management help can be used. The second one is by applying it independently after the execution of one (or more) functions, and utilizing as data particular exodirected messages issued by such functions. Functions must be programmed to issue such messages by the system analysts, to whom the operations researches must tell what information such messages should contain. Then the mathematical analysis of such data can be performed automatically by the computer as soon as all data are available (in which case the operations research people will write the program and the system analysts will schedule its runs) or by the computer as independent task started by human interventions, or can be done with pencil and paper.

In any case, this latter type of application of operations research techniques is based on information which can be viewed as a selective summary from the data base. Hence we can name its self-contained environment a <u>summary self-contained information</u> environment. Examples of self-contained processes which commonly occur on summary environments are simulations, linear programs for optimization of continuous production processes, statistical analyses for reporting in connection with long range planning purposes, etc.

3.5 <u>Parametrically modified executions of functions</u>

Occasionally it is wise to consider running a function faster than usual. This requires an abstract, as opposed to the full original, or the file involved. This can easily be done within the framework disucssed here as follows

- The <u>abstract</u> file (s) is (are) generated or regenerated as a sequence of transfer messages every time the function is executed with the full file.
- Such abstract files are used as normal files to execute functions. Functions based on abstract files update the abstract file involved: however, such updated abstract files are to be viewed (and treated) as sequences of transfer messages.
- Each time the function is executed with the <u>full</u> file, all of the abstract files deriving from it (each of which will perhaps have been updated many times) will be put into the input batch for the function. The program of the function must take care of entering into the full file all of the recent modifications carried by each one of the abstract files deriving from it. Then the old abstract files are eliminated.

More in general, in order to meet requirements concerning fast responses based on limited information, one can condiser alternate, reduced, i.e., parametrically modified versions of some functions. This delicate matter is totally in the hands of system analysts (not of managers) and involves programming the functions in a way to accommodate parametric modifications.

Bibliography

1. Leavitt, H.J. and whisler, T.L., Management in the 1980's, Harvard Business Review, Nov. - Dec. 1958.
2. Anshen, M., The Manager and the Black Box, Harvard Business Review, Nov. - Dec. 1960.
3. Burlingame, J.F., Information Technology and Decentralization, Harvard Business Review, Nov. - Dec. 1962.
4. Dearden, J., Can Management Information be Automated?, Harvard Business Review, March - April, 1964.
5. Brooks, F. B. and Iverson, K. E., Automatic Data Processing Wiley, 1964.
6. Gregory R. H. and Van Horn, R. L., Business Data Processing and Programming, Wadsworth, 1965.
7. Lombardi, L. A., Hardware requirements of multi-access and multi-computer systems, (unpublished)
8. Lombardi, L.A., Theory of Files, Proc. 1960 Eastern Joint Computer Conference, New York, N.Y.,(1960) paper 3.3, 137-141.
9. Lombardi, L. A., Nonprocedural Data System Languages, Invited Paper, Pre-prints of papers presented at the 16th National Conference of the Association for Computing Machinery, Los Angeles, California, (1961).
10. Logic of Automation of System Communications, J. Machine Accounting (13) 4 (1962), 18-28.
11. Mathematical Structure of Non-Arithmetic Data Processing Procedures, J. Assoc. Comput. Mach. (9) 1 (1962), 1936-159.

12. On the Control of Data Flow by Means of Recursive Functions, Proc. Symp. "Symbolic Languages in Data Processing," International Computation Center, Roma, Gordon and Breach, 1962, 173-186.

13. On Table Operating Algorithms, Proc. 2nd IFIP Congress, Munchen (1962), North Holland Publishing Company, Amsterdam, 230-232.

14. Les Perspectives du cacul automatique, SCIENTIA, (in Italian and French), IV series (57) 2 and 3 (1963)

15. Mathematical Models of File Processes, Atti. Sem. Mat. Fis. Univ Modena, (12), 1963, 173-214.

16. A General Business Oriented Language Based on Decision-Expressions Comm. Assoc. Comput. Mach. (7) 2 (1964), 104-111.

PART II. NEW TECHNIQUES OF IMPLEMENTING DATA-BASE INFORMATION SYSTEMS.

4. Implementation and Programming Tools

4.1 Sorting and interfacing

The common technique for executing a function of a collinear data base information system in a way such that every record is available in conjunction (i.e., during the execution of the same phase) with the availability of all endodirected messages pertaining to it, is based on requiring that both the file and the batch of all input messages are sorted in sequence by increasing value of the common key. Hence all files must always exist in sorted (sequenced) form. A file is never sorted (except at most when it is first originated). Hence, the batch of input messages needs to be sorted before the execution of a function.

Sorting can be handled by the computer by means of techniques widely covered in the literature (see for example the ACM Journal and ACM Communications of the last eight years). We will not here explain nor require the reader to be familiar with such techniques. We will instead restrict ourselves to some peripheral considerations about them on which the implementation plan proposed here pivots.

Sorting on a magnetic tape or magnetic disk computer is performed by a method called merging. The sorting process is divided in two successive steps, namely external sort (also called pre-sort) and internal sort.

Let us now consider the case of a computer using magnetic tapes. Then the external sort consists of

- Reading into the memory of the computer strings as large as possible (e.g., 50 messages or so) of the raw (unsorted) input batch. This input batch consists of messages coming from different sources. The transfer messages of the input batch will be on a magnetic tape unit, referred to as sort input unit, while the exogenous ones may be either also on the sort input tape unit, or on cards placed in the hopper of the card reader (s), punched paper tape put on an appropriate reader, magnetic or optical ink checks placed in a suitable recognizer, etc.
- Sorting of such strings into sequences within the memory of the computer.
- Writing such sequences on two sort output tape units, on a flip flop basis, i.e., one of the two output tape units will be used for writing the first, third, fifth, etc., of such sorted strings while the other one will be used for the even-numbered ones.

So three tape units are needed for external sorting. After its execution the tape(s) mounted on the sort input unit is of no use so that, unless its contents should be saved for security against breakdown, it can be used as a blank tape.

An internal sorting consists of a sequence of iterations (passes), each of which consists of copying the available successions of sequences from tapes onto other tape units by merging couples of such sequences into a single, longer one. This yields eventually to the reduction of the two original successions of sequences formed on the sort output units into a unique sorted sequence, which is written on a <u>sorted messages</u> (tape) <u>unit</u>. In order to do so one needs at least a total of three tape units (better 4), inclusive of the sort output units and the sorted messages unit.

Originally external sorts were designed to use four tape units. Comparatively recently more sophisticated techniques were invented (i.e., <u>Fibonaccian</u>, <u>cascaded</u>, <u>polyphase</u> sorting) which limit requirements to three units with a comparatively small marginal loss of efficiency (whose magnitude, in any case, depends strongly on some particular features of the computer hardware involved). However, for reasons that we will see in a short while, a data base information system needs always at least four tape units. So that there is no point in using 3-unit sorting, no matter how small the marginal loss is, because this would imply leaving a tape unit idle.

On the other hand, it is well known that both external and internal merging efficiency can be improved by slightly extending the algorithms and using additional couples of tape units. (The above is called <u>2-way merging</u>, while we can have more efficient <u>n</u>-way mergings (<u>n</u>>2) which need <u>n</u>+1 tape units for external sorting and 2<u>n</u> tape units for internal sorting. The marginal gains in efficiency obtained by increasing <u>n</u> depends on the computer hardware, but is generally small. Since this involves adding new (very expensive) tape units, the marginal profitability of such augmentation is generally negative if the additional tape units are not utilized other than in sorting. But we will see in the immediate sequel that, apart from sorting, a data base system cannot possibly use more than 4 tape units.

In conclusion, the present study will lead to the discovery of the fact that complex 3-tape or multi-way sorting methods, once considered quite important in information processing, have really no significant place in the field, while a balanced collinear information system should always rely for its sorting needs on a clean, simple, 2-way merge.

4.2 Execution of functions with tapes and the 4-tape principle

The computer of a data base information system executes only two kinds of operations: sorting (external and internal) and functions. Let us study the latter.

Here one tape unit is needed for as old file unit, and a second one as updated file unit. A third tape unit is needed to carry the sorted messages coming from the sorting process: a single unit is sufficient, but also necessary because, since messages have to be sorted, they must come to the function written on a tape (as opposed to a deck of cards, etc.). We will now call this unit (which physically coincides with the sorted message unit of the preceding sort, but plays now a different role) function input unit. Card readers and similar input devices are idle during the execution of a function (except for monitoring purposes). Some of the exodirected outputs can be directly sent out via on-line telegraph wires, teletype, etc., or printed by on-line printer(s) while they are being generated. But all of the transfers generated by the function must go on a further tape unit. In addition, all streams of exodirected messages which cannot be directly issued (e.g., there might not be enough printers available) also have to go on tape units. Now conventional system analysis here suggests using a separate tape unit for each stream of output messages. This is extremely expensive.

Instead, a single tape unit (the fourth one, that we will call function output unit) should be used for batching all streams of transfer and exodirected messages, except those which can be dispatched on-line. Each message should contain a code (stream code) identifying the stream to which it belongs.

The question now is: how do we split such streams apart? The next operation of the computer after the execution of a function, say f_1 is always first an external, then an internal sort preluding to another function, say f_2. The reel(s) containing the output messages of $\underline{f}_1$ will be batched together with all other reel(s) containing input messages for $\underline{f}_2$ coming from sources other than $\underline{f}_1$ (as a matter of fact, a reel already partially filled with transfer messages can in some cases be used for receiving the output of $\underline{f}_1$). Then the pre-sort of $\underline{f}_2$ must build the above discussed strings selectively, in the sense that, among the messages coming from tape, it will consider only those whose stream identifier directs them to $\underline{f}_2$.

Now (and this is the focal point of the new approach) pre-sort uses only 3-tape units, so there is a fourth one sitting idle. So, all and only those endodirected messages coming from the input tape unit which are not directed to $\underline{f}_2$ will be copied on this fourth tape unit (named floating unit) which hence, at the end of pre-sorting, will contain all endodirected messages which are still meant to be around after the execution of $\underline{f}_2$. Then, when $\underline{f}_2$ is executed, its output messages will be simply batched together (sharing reels whenever desirable and feasible) with whatever was the contents of the floating unit at the time of the preceding pre-sort. By means of this continuous reduction process, all transfer information is kept together as a compact, unified, minimal body of data.

Moreover, all exodirected output units (printers, telegraph connections, etc.,) not directly utilized during pre-sorting. So, during pre-sorting, we can take advantage of the situation for performing selective printing or dispatching as many streams of exodirected data which were found on the sort input unit as possible, up to the point of loading all suitable exodirected output units of the computer. All those exodirected messages which cannot be issued because all such units are tied up will be copied on the floating tape, waiting for a further chance of being printed or dispatched during a forthcoming pre-sort.

In conclusion, the pre-sort program should handle simultaneously with pre-sorting, hence at no appreciable additional cost (in terms of speed of operation) with respect to straight pre-sorting, all of the following operations:

- Converting messages which are not on magnetic tape to tape format
- Proving the exogenous input according to management's specs
- Proving the exogenous input according to specs established by system analysts
- Selection of messages to be pre-sorted
- Pre-sorting
- Selective printing and dispatching of as many streams of exodirected messages as the computer can handle
- Copying on the floating tape unit all input messages which are neither pre-sorted nor printed

This complex of simultaneous operations will be referred to as external interfacing.

So we showed that not more and not less than 4 tape units are needed for the efficient implementation at maximum parallelism of a collinear base system using magnetic tapes: This is the 4-tape principle.

The pre-sort program, which should be a parametric program (generator) supplied by the computer manufacturer, should be designed to handle those seven operations. No pre-sort program in existence today does that: in fact, even the most sophisticated ones only handle external sorting and card-to-tape conversion.

This is a sympton of the gap that there is between proper and current design of management information systems.

The dynamic allocation of operations such as printing or dispatching of exodirected messages to those functions or pre-sorts which have appropriate hardward available is a job of the systems analyst. He must decide under three kinds of constraints, to wit:

. The fact that dispatching units and especially printers tend to be expensive to buy and maintain, and

. In most cases printers and some other dispatching units can only handle one stream of messages at a time (because for example different reports cannot be printed on the same ream of paper)

. The response requirements set by management for exodirected outputs should be met

In very many cases proper systems analysis allows to require only one printer in almost all information systems, no matter how sophisticated. In addition, this study shows that by distributing harmonically the load among devices and steps of the operation, every device can be utilized to full capacity and the response time can be minimized: all this using minimum equipment.

A further indication of this study is that complex and expensive techniques emphasized by conventional analysis, such as multi-processing (i.e., executing independent programs simultaneously), not to speak about the traditional offline card-to-tape and tape-to-print conversion, have little role in management information systems.

Frequently some exodirected messages on the floating unit need sorting before printing: this is often the case for long reports. This can be handled either by scheduling ad hoc sorts (which are run like all other sorts) or, when possible, sorting them together with the input for a function. (Such function does not have to have the same key connotation as the output stream.) The trick here is to print (without putting it on the sorted input unit) such exodirected messages during the last iteration (pass) of the internal sort, by taking advantage that, during internal sort printers and dispatching devices are idle. Quantitative considerations on sorting times indicate that the first approach allows for faster sorting, while the fact of utilizing an expensive device such as a printer at the only time when it can not be utilized otherwise makes the second approach marginally convenient in many cases.

Moreover, the need of better utilizing the printers might in some cases suggest the opportunity, during pre-sort, of [illegible]ting some streams of exodirected messages on the output units even if they do not need sorting. Then the internal sort program can print them out selectively (without copying them back on tape) during the very first pass. Or else the internal sort program can selectively print out different streams during different passes, thus maxizing the use of the printer and reducing average response times. The complex of simultaneous operations consisting of an internal sort and the extension thereof suggested in the last two paragraphs will be designated as internal interfacing. A sequence of two steps, the first being an external and the second an internal interfacing, will be referred to as interfacing.

However, sorting programs able to handle the efficient approaches implied by interfacing have not been written so far by any computer manufacturer, although there is no theoretical difficulty in developing them.
In many cases one can justify a fifth and a sixth tape unit as spare parts, to take over when one tape unit is down for maintenance. In some other cases, especially in connection with large, multi-reel files, one might want to double some tape units so that one of them can be used by the computer while the operators change reels on the other one. For either use there are never more than 4 units operating at any time. This is actually not in derogation to the 4-tape principle, if we consider no longer

physical tape units, but rather logical tape units, whereby each logical unit consists of several physical ones, of which not more than one operates at any time. But except for the case when tape units in excess of four are used in either of these two ways, their presence in data base information systems is by and large redundant, and reveals bad system analysis (although some salesmen present it as a status symbol).

4.3. Execution of functions with disks and the 2-disk principle

Let us now switch to the case of a collinear data base system implemented by using magnetic disks (or for that matter any other random access device) as opposed to tapes. We will assume that disks (or stacks of disks) are removable from their units: otherwise it would be difficult to go far in the way of proper systems analysis for data base systems.

A trivial approach to comparing the relative merits of disks and files is to conceive a system operating as described in the previous sections, except that the four tape units are replaced by disk units. Since the cost of such devices are comparable, the differences in cost/performance ratio are small, debatable, and change with the technological advances. The main claimed advantages of disks are that they do not wear, have no skewing problem on the coding channels at reading time because of their rigidity (and hence provide an advantage which can be exploited alternatively in terms of higher reliability, higher reading speed, higher density, or lower cost of reading circuitery), and they are easier to operate manually. The advantage of tapes is that the capital investment in reels for keeping information stored is lower by a factor of 10 or so. However, the latter is generally a minor component of the total cost of an information system, so that most analysts now suggest having disks in any case, and using tapes only to store bulky dead records, i.e., records which are kept for a long time with a low frequency and/or a low probability of usage (e.g., records used for audits, records kept for breakdown emergencies or for legal or taxation reasons, etc.). Conversion from disk to tape or vice versa can be handled with a single tape unit or (generally more conveniently) by employing a Service Bureau.

These are small margins provided by small mechanical differences. However, the new approach indicates that there are large margins in distinct favor of disks provided by the possibility of a better design of collinear systems. In fact, the four tape principle applied to disks reduces to a two-disk principle. That means that only two disk units (as opposed to four tape units) are necessary and sufficient for implementing a collinear data base information system. The reason for this difference is that to a large extent we can take advantage of the random access features in order to have different areas within the same disk (or stack of disks on the same unit) carrying out the operation that, using tapes, had to be performed by different units.

The first thing that one can think of doing along this line is to try to use only one disk unit, dividing the areas of disks mounted on it in four sections which correspond to the above four tape units. This works only when the data base is so small that it can be completely permanently contained (all files) in a section of the same disk, while space should be left over for messages and sorting areas (analogous to blank tape reels).

Otherwise, one or more disks (we will refer to a solidal stack of disks simply as a disk) should be assigned to every file. Each of such disks may be at least half blank, so that it can be updated by copying its contents between areas on a see-saw basis (although a well known technique called chained addressing allows to easily overcome the waste of disk space and copying time implied by such approach: the latter approach is preferable to the first).

Let us consider the execution of a function $\underline{f}_1$. If we had only one disk unit, the sorted messages must be on the same disk unit as the file. The output messages should also go on to the same disk unit, and consequently on the same disk. Consequently, they would remain attached permanently to the updated file. The next step (sorting for the function $\underline{f}_2$) can be still performed on one unit, provided that there is some blank space left on the disk. But when we come to executing the function $\underline{f}_2$ the disk must be replaced by the one containing the file relative to $\underline{f}_2$, and hence the sorted input to $\underline{f}_2$ would no longer be available to the computer. So a single disk unit is not sufficient.

But two are. Let us assume that we have two disk units and let us denote them D_1 and D_2, respectively. Then, during the execution of any function, D_2, will carry disk (or disks, in sequence) which is (are) entierely and exclusively devoted to carring the file involved (both in the old and updated version). So D_2 performs the work of the old file tape unit and the one of the updated file tape unit. D_1 is devoted to carrying the sorted input to the file as well as all of the information which, using tapes, would go on to the function output tape. So D_1 performs the work of the function input tape unit and the one of the function output tape unit.

During sorting, a blank disk can be mounted on D_2. The external sort would utilize D_2 like it would use the two sort output tape units, while all information which would go on to the sort input and floating unit is recorded on the disk of D_1 (again, adoption of chain addressing can save both on disk space and copying time). So D_1 performs the work of the sort input tape unit and the one of the sorted messages tape unit. (Internal sorting can be handled by using arbitrarily both D_1 and D_2, except that the final sorted sequence must be placed on D_1.) So, at the end of sorting, D_2 carries a blank disk again, which can be replaced by a disk containing the file relative to the function to be executed next. This is the operating plan of the <u>2-disk principle</u>.

A feature of this operating plan is that the disk on D_1, which carries all of the interface information, does never need to be removed from its unit.

As it was observed by one of our graduate students at M.I.T. (whose name, unfortunately, we failed to record), one can still operate with just one disk unit without losing generality provided that the stacks of disks are designed to be broken down physically in two pieces, which would correspond to D_1 and D_2, respecti . Moreover, since the disk on D_1 never needs to be removed during normal operation, one can think of building a disk unit with removable disk stacks similar to the IBM 1311 or 2311, except that only the upper half (perhaps more than half), corresponding to D_2, of the <u>Disk Pack</u> would come out when the operators twist and pull the handles, while the lower part, corresponding to D_1, would remain on the unit. Such device has no reason of being more expensive than one regular IBM 1311 or 2311, while it would replace two such devices. This minor change in hardware design would allow to handle a complete collinear data base system with just one disk drive. We may call this result <u>single-split principle</u>. This would cut in half a quite important component of the cost of an installation, namely the one relative to disk (or tape) units. This result shows the usefulness of the new approach to systems analysis in suggesting profitable hardware design features.

The 4-tape and 2-disk principles allows to drastically reduce capital investment, and also provides a way to evaluate for budgetary purposes the cost of a collinear information system. However, they hold only for implementations at maximum parallelism, which, as discussed earlier, have the advantage of broadening the scope of the supply of information and reducing both response time. More precisely, the 4-tape and 2-disk principles are the specialization to implementation with present day equipment of the more general maximum parallelism principle.

In conclusion, information systems can be implemented inexpensively only conditional upon their being fast and providing a good service. Between inexpensiveness on one side and cost and effectiveness of service on the other there is a mutual dependence, not a trade-off, as conventional systems analysis suggests. And the same is true also for on line implementations, discussed in the chapter 5. Of course, such seemingly paradoxical situation is typical of a time of technological change, where high cost slowliness and limited service happen to be all three associated to the old technology and not to the modern one.

And let us also recall the two kinds of operations that the computer does:

1) Executing functions
2) Interfacing

Now the question is: what kind of software tools, (i.e., programming, languages, compilers, generators, etc.) does the system analyst have or should he have in order to implement a collinear data base system?

None, except general understanding, is required for routing. As far as interfacing is concerned, he should be enabled to use program generators supplied by the computer manufacturers, of the type of the currently available sort generators, but able to handle all aspects of interfacing (some of which are occasionally called <u>report generating</u>).

Writing, changing and extending programs for functions can be handled today only by using machine language programming, or better simplifications thereof commonly called <u>autocodes</u>. Also COBOL type languages IBM Commercial Translator, FACT) can be used, since they are actually not much worse than autocodes in describing procedures, and have the remarkable (and only) good feature of allowing for a compact, efficient description of the layout of the files and their records and of the streams of messages, <u>(data division)</u>, whereby such data description is separate from the program of the functions (<u>procedure division</u>).

ELEA 9003 computer under the name N1 (for <u>non-procedural</u> 1). A full account of N1 would unduly lengthen this paper. We will instead devote the remainder of this section to explain how N 1 is related to the systems analysis approach here described. We will assume that the reader is familiar with [16] : otherwise we suggest that he skips the remainder of this section.

This approach to system analysis can, at least theoretically, be implemented with conventional languages (i.e., COBOL, FACT, or autocodes). However, such implementations would be costly program, would be rigid, and would bar swift interplay between the evolutionary world of management's policies and their implementation on computers, Still, the advantages in terms of reduced amount of hardware and shortened response time could be achieved without N1.

Conversely, N1 would help even in the framework of conventional systems analysis. Its advantage would be to add flexibility and facilitate the task of programming. But its full potential comes to light only in connection with the new systems analysis.

To explain this, it is in order to recall some features of N1, namely the way the separation between policy and system is obtained.

The <u>compiler</u> (really a <u>generator</u>) for this new language N1 contains the <u>quid commune</u> to all functions of all information system, namely a universal model of the flow of data. The identification of this model has been an instrumental discovery, without which the new language would not be possible. So the system analyst who programs a function in N1 is completely relieved from any burden concerning writing pieces of code which command entering or issuing data into or from the computer (this burden accounts for most of the difficulty in programming a function using conventional languages like antocodes or COBOL). Among other things, this implies that, when using N1, high parallelism is no longer a primary source of programming difficulties.

But if the analyst has available no other tools but these the programming of functions is extremely difficult, especially because

- The parallelism principle requires handling within the same functions a large number of streams of messages. This brings up quite complicated problems of data flow coordination.
- The implementation of increasingly complex and diversified policies designed by different branches of management (with or without mathematical help) and of answering mechanisms to increasingly complex and diversified questions, also asked by different branches of management, involves two difficulties:

  a) Reducing such policies or questions, which are presented as statements of the desired result (i.e., <u>non-procedurally</u>), to time-sequenced programs or procedures to achieve such results and

  b) incorporating such new pieces of program into the large programs of functions without losing a synopsis of the interrelation-ship (in terms of <u>flow of control</u>) between the different pieces
- To the extent to which management and the physical environment are <u>coextensive</u> with the information system (i.e., to the extent to which the latter is not relegated to trivial accounting), frequent unanticipated changes and extensions of the scope of the functions is necessary. With present tools this involves reprogramming of all of a function most of the times that a change occurs. The lead time and cost implied by this are prohibitive

So a new tool is needed in terms of a really management-oriented <u>computer language</u> and a <u>compiler</u> for it. The philosophy underlying such a language has been developed and discussed in previous papers see 8,9,11,12,13,14 and for an informal discussion, especially 16 , and partially implemented[1] for the Olivetti

[1] Dr. M. Pacelli, Mr. G. Palermo and their associates

In the new language the program of any given function (corresponding to a procedure description of COBOL) consists of a set of independent and unrelated (i.e., modular) statements, divided in two groups:

a) Flow control predicates. Each flow control predicate is an expression whose value can be true or false (i.e., a predicate). There is one flow control predicate associated to the updated file and one associate each stream of output messages. Its meaning is follows:
"a record or message should be issued into the fi stream associated to the predicate during all phases (and only those) for which the predicate has the value true." Decisions as to whether to issue a record (i.e., keeping or not keeping a record in the data base, or entering a new one into it) or as to issue or not issue exodirected messages is an important kind of management decision, while similar decisions concerning a transfer message is an important kind of system analysis decisions: hence their specifications are left open as part of the language, as opposed to be standardized in the compiler.

b) Field declarations A field declaration is an xpression whose value is the contents of a field of an outp message or record. There is one field declaration associated with each field, while fields common to several output messages and an output record may share a single field declaration. What exactly to put into a field is an important management or system decision, so this again is part of the language, not of the compiler.

Each such statement is not a program, but an expression. Each of them fully represents a particular facet of the policy

of the organization in terms of a declaration of
the result that it should issue. Each of them is completely independent of any other one, to the point that they can be fed to the compiler in arbitrary order: so single policies can be added, modified or cancelled without any concern about the others. Each such statement represents a policy as a declaration of a result, i.e., directly in managerial decision-making terms, as opposed to representing it as a procedure to obtain such result, i.e., in computer terms. The procedure to work out the specified result as developed by the compiler. Multiple conditional alternatives can be contemplated within a given policy by using a flexible and easy technique called conditional function. Different statements expressing decisions made in different decision loci can be written by different managers, even if they do not talk to each other.

The substantive idea here is that any managerial decision or request of information, as well as any stimulus from the environment, can be expressed in terms of either having or not having a certain record in the data base or message, or in terms of the contents of fields of records or messages. This is necessary and sufficient to express any decision or request. A statement is the most direct way managers can express decisions or request. So a set of such statements is all that should go into the representation of a function: everything else, which is not policy, but routine or procedure pattern is permanently encoded in the compiler.

So compactness and ease of formalization of policies, as well as flexibility and expandibility, are very high in N. . A change of policy can be implemented instantaneously: all it takes is writing or rewriting a statement (or a part thereof) of the particular function involved, without any concern about the other statements of the same function.

On the other hand, this reduction of the programming effort to a simple expression of policies, eliminating from the explicit representation of functions everything which is not **policy**, is the theoretical ceiling of what can be done in the area of management languages: any attempt of further reduction would imply freezing into the compiler aspects of policy decisions, i.e., giving to the information system a bias toward certain policy decisions, i.e., giving to the information system a bias toward certain policies. But, as we said, policy matters should never be of concern to the system analyst.

Conversely, our new approach, with the help of N1, re[illegible]ves the systems analyst from any management interference the way specified policies are implemented (i.e., reduced to procedures).

In conclusion, all that is needed (besides understanding of the problem) as tools for implementing an efficient collinear data base system are two packages of software, to wit:

. An interfacing generator, discussed in section 4.1
. A compiler for the language N1 (on for a similar language).

This is the <u>two-software package principle</u>.

Most of the program generators in existence today cannot be directly used in implementing the new systems analysis approach. In particular, no progress is to be expected in the area of management information systems until interested parties will tie their valuable programming talents on the development of isolated parametric programs such as <u>report generators</u> (separate from both functions and interfaces!), <u>file updating generators,</u> etc. The emphasis on this type of tools, as well as the shallow glamour of COBOL-type compiler languages, epitomizes the current state of confusion and misinformation in the whole area of management information systems.

When functions are represented in N1, an executive has two ways of conveying a request of information or a notification of policy to the information system, namely:

- . Writing an exogenous message directed to specified functions. The fields of such messages may contain simple parameters, but they may also contain full-fledged statements written as expressions in the N1 language (parameters, variables or constants, are special cases of expression in N1). In the latter case the N1 compiler will interpret at run time the contents of such fields, and execute them (although a better solution consists of letting the interfacing program take care of compiling such statements). Having additions, deletions or changes performed on particular statements the programs of specified functions.

In turn the system analyst will apply such information coming from management to the functions affected, and design the appropriate routing. For this purpose he must analyse, for each case, which are the files directly or indirectly involved. So he must learn to conceive every record on endogenous messages as a function of a set of files (a subset of the data base) and a set of streams of endodirected messages, and determine such sets for each use.

With this tool (N1 and it underlying system organization model) management and information system can interact.

- . Instantaneously
- . At a high and increasing levels of complexity

Or, in other words, they can be coextensive. So the new approach has the potential for removing the obstacles, which are holding back the information technological revolution in management.

5. On line data base systems

5.1 Typical organization

The new organization plan for collinear data base systems allows to cut by order magnitudes the response time of conventional systems. The lower bound of the response time that can be reached through this plan using present hardware is of the order of hours and perhaps tens of minutes, while the one relative to conventional, low parallelism systems is by and large of the orders of months or weeks, and only in few cases of days. (This implies that the new approach to management systems analysis has some potential for reviving the usage of collinear systems.

The lower bound to the response time in collinear systems depends on the time necessary to scan a file. With present technology this is rarely less than several minutes. But there are cases where the response time requirements are of seconds: as we said, such cases are mainly in connection with dry information processes.[1] For these cases the clean, uniform and inexpensive collinear approach is no longer sufficient, and we have to resort to on-line data base information systems.

In on-line systems

- Records of the data base can be retrieved (accessed) at random, that is, accessing of one record does not imply any action on the other records of the file to which it belongs. This implies that magnetic tape units cannot be efficiently used; instead, for supporting the data base and should use random access storage devices like magnetic disks, drums, delay lines, magnetic card libraries, or any of the frontier devices which are currently under development (for simplicity we will always refer to all such devices as disk units).
- All of the data base (as opposed to single files) should be accessible to the computer at any time. This implies that all of the data base must be permanently mounted on disk units. So the possibility of dismounting disks from the respoecive units (like in the IBM 1311 or 2311), which is instrumental in collinear systems, is actually of minor relevance to on-line systems. (this explains the puzzled and puzzling incoherence of the reaction of the market to the IBM 1311 and 2311).

1)

1) The determination of response time requirements for messages or classes of messages rests with the systems analyst. The timing paradox is the only theoretical help that he has for this purpose. And such help is in terms of an enlightening philosophy, not of a quantitative method. The determination of such quantitative methods, compatible with the timing paradox, would be very helpful, and we suggest it as a possible area of research.

The subdivision of the operation of the information system into functions still holds. However, all functions are no longer executed separately. Instead, their executions are intertwined. More precisely, the computer executes in sequence complete phases of different functions which are related by transfer messages.

For illustration, let us refer again to the simple example of the wholesale company, whose data base has at least three files-inventory, customers and suppliers. If the system is on-line, the routing of an exogenous order will first call for the retrieval of the corresponding stock item record from the inventory file. The phase relative to the inventory function is executed by and large like in the previous case of the collinear system, and a transfer message to the customer file is prepared.

However, this transfer message actually is never issued. It never gets out of the computer memory. After the execution of this inventory phase, instead of going to another inventory phase, the computer shifts to the customer function, using as input the above virtual transfer message, and executes a customer phase, which may yield the completion of the order of shipment and related exodirected messages. If the order cannot be completed because of policies encoded in the customer function, another virtual message directed to the inventory function is issued, and the computer shifts back to executing a phase of the inventory function.

An analogous sequence is performed for transfers to the supplier's function for re-ordering or other purposes.

5.2 Common language for collinear and on-line data base systems

So, beyond a shallow appearance, the differences between collinear and on-line information processes is neither in the functions nor in the routing patterns, but in their interfaces. Functions and routing patterns are identical, and, consequently, they do not need reprogramming or redesign when one shifts from collinear to on-line.

The differences in interfacing between collinear and data base systems boil down to the following:

. Transfer messages are virtual, as opposed to being physically issued by the computer

. There is no need for sorting, except for the case of sequenced reports (which are exodirected streams of messages)

An immediate consequence of this is that the language for programming functions for a collinear system is good also for an on-line system, and viceversa. The widely held opinion that there is a need for a special language for representing functions of on-line systems has no substance in the light of our new approach systems analysis.

But this holds only for the language, not for its implementation. The above universal model of data flow is characteristic of functions executed collinearly. For executing them on-line, we need a compiler which contains permanently recorded the sequencing scheme epitomized by the above example of the wholesale company treated on-line. Such compiler has not yet been fully studied, although some basic concepts for building it are contained in section 2 (table operations) of reference 11 . see also 13 .

On-line systems cannot handle the problem of printing exodirected messages on the basis of getting a free ride on the sorting programs, because there are no sorting programs. On the other hand, the fact that the usage of reams of paper gives to printers an inherently collinear characteristic might require some measures, although the habitat of on-line systems naturally relies less on sequential printing than on immediate dispatching of exodirected messages. Such measures, which specifically effect the handling of those streams of low-priority messages which need to be printed although all printers are tied up by different streams when these messages are first generated, can be taken in terms of

- . Physically issuing such messages as punched cards or so, and designing functions which have no other purpose but printing them. Phases of such functions can be activated by simply feeding back such cards into the computer whenever a printer is available.

- . Recording such messages on disks, and again designing special printing functions as above. In this case the compiler (or better, the part of the software called <u>supervisor</u>), should automatically activate such printing functions by entering a message from the disks each time both there is a printer available and there is no exogenous message with relative priority which calls for service.

5.3 <u>Hardware for on-line systems</u>

Finally, we might add that present day hardware is [illegible] fully adequate for the implementation of on-line data base systems, except on a very small scale. Attempts to build large systems on the basis of present hardware yields unwieldly cost figures.[2]

---

2) The order entry system of Westinghouse in Pittsburgh, Pa., the SABRA reservation system of American Airlines in Briarcliff, N.Y., and the on-line job shop scheduling system of Lockheed in Sunnyvale, Cal., are examples of such high cost performance ratio.

So we may ask whether on-line systems built with present hardware have a built-in element of much higher extra cost.

They are inherently more expensive for just one reason: the fact that they need all of the data base continuously available for access, so that they need a large capacity complex of random access storage devices.

However, on-line systems built with present hardware have a built-in element of much higher extra cost, which depends on the fact that they have a limited number (one or two) or independent accessing devices for every disk unit. So they can handle only a limited number of accesses to the data base per unit of time, and the internal processing capacity of the computer can be utilized only to a small percent.

However, this reason of extra cost is not inherent to on-line systems. It just depends on the fact that appropriate hardware i not available today. The reason why it is not available is not technological, but it just depends on the lack of indications supplied by system analysis. This topic has been analyzed in a different paper 7 , where design specs of random access storage devices for the data base of on-line management information systems are given. So there is no point in rediscussing it here.

5.4 <u>The functions polyhedron</u>

An intuitive representation of the system organization plan proposed here (both collinear and on-line) consists of viewing the information system as a polyhedron, each face of which represents a function. The edges of this polyhedron represent the interfaces (sorting, selecting, printing, dispatching, etc.,) between functions. Thus one can plan the sequence in which the functions are executed by means of a closed line (<u>route map</u>) on the polyhedron, which may cross the same face several times. Each such crossing, denoted an execution of the corresponding function. Each such execution should be preceded by the process of interfacing, the batch of all messages issued by all functions whose faces have an edge in common with the one considered.

6. Final Remarks

So far we have discussed efficient ways of planning automatic management information systems within an organization. The next step is to study how we can extend automation to the communication without routine human intervention between information systems of different organizations. This would eliminate the need for human recording and keypunching of messages which were originally issued by machines, which is slow, unreliable and expensive.

The problems involved here are not simply the ones of compatibility of bitcoding of characters, nor the ones of universal acceptance of a standard set of magnetically or optically recognizable characters. The solution of the last mentioned two problems is a necessary, but by far insufficient condition for implementing automatic inter-system communications. The most difficult and yet unsolved problems in this area are the ones of semantics, i.e., those referring to the contents of the fields of messages. More precisely, there should be an established way of communicating identifiers of corresponding parties and codes of objects to which transactions refer, as well as a flexible but standardized way of formatting inter-organization messages.

However, this problem and all of its implications and organizational difficulties have already been discussed in another paper (10), so that there is no point in rediscussing it here.

Bibliography

1. Leavitt, H. J. and Whisler, T. L., Management in the 1980's, Harvard Business Review, Nov. - Dec. 1958.

2. Anshen, M., The Manager and the Black Box, Harvard Business Review, Nov. - Dec. 1960.

3. Burlingame, J. F., Information Technology and Decentralization, Harvard Business Review, Nov. - Dec. 1962.

4. Dearden, J., Can Management Information be Automated?, Harvard Business Review, March - April, 1964.

5. Brooks, F.B. and Iverson, K. E., Automatic Data Processi[illegible], Wiley, 1964.

6. Gregory R. H. and Van Horn, R. L., Business Data Pro[illegible] and Programming, Wadsworth, 1965.

7. Lombardi, L. A., Hardware requirements of multi-access and multi-computer systems, (unpublished)

8. Lombardi, L.A., Theory of Files, Proc. 1960 Estern joint Computer Conference, New York, N.Y., (1960), paper 3.3., 137-141.

9. Lombardi, L.A., Nonprocedural Data System Language, Invited paper Pre-prints of papers presented at the 16th National Conference of the Association for Computing Machinery, Los Angeles, California, (1961).

10. Logic of Automation of System Communications, J. Machine Accounting (13) 4 (1962), 18-28.

11. Mathematical Structure of Non-Arithmetic Data Processing Procedures, J. Assoc. Comput. Mach. (9) 1 (1962), 1936-159.

12. On the Control of Data Flow by Means of Recursive Functions, Proc. Symp. "Symbolic Languages in Data Processing," International Computation Center, Roma, Gordon and Breach, 1962, 173-186.

13. On Table Operating Algorithms, Proc. 2nd IFIP Congress, Munchen (1962), North Holland Publishing Company, Amsterdam, 230-232.

14. Les Perspectives du cacul automatique, SCIENTIA, (in Italian and French), IV series (57) 2 and 3 (1963).

15. Mathematical Models of File Processes, Atti. Sem. Mat. Fis. Univ. Modena, (12), 1963, 173-214.

16. A General Business Oriented Language Based on Decision-Expressions Comm. Assoc. Comput. Mach. (7) 2 (1964), 104-111.