Decompostion of information systems for production management

Decomposition of Information Systems for Production Management

H.J. Pels and J.C. Wortmann
Department of Industrial Engineering and Management Science, Eindhoven University of Technology, Eindhoven, The Netherlands

This paper approaches the issue of decentralization and decomposition of information systems from two angles, viz. from an organizational and from an infological point of view. Current information systems tend to become more and more integrated. However, this integration causes organizational complexity, which, in turn, becomes prohibitive for organizational change. Thus, there is a need for decomposition of the information system from an organizational point of view. A strategy for such a decomposition in a production environment is given.

Keywords: Information Systems, Production management, Distributed Processing

Henk Jan Pels is a member of the scientific staff of the Informations Systems Group in the Department of Industrial Engineering at the Eindhoven University of Technology. After he received his degree in electrical engineering at the same university in 1971, he joined Philips Industries, where he was engaged in the development of database systems for the bussiness as well as the manufacturing area. In 1981 he moved to his current position, where he is researching and teaching the use of data models and database design, especially for production control applications.

Johan C. Wortmann studied industrial engineering and management science at Eindhoven University of Technology, The Netherlands. He has been active in the development of information systems for production/inventory control since 1973 and wrote a doctoral thesis on the subject. He has been involved in a number of practical applications, both in component manufacturing and in assembly operations. He worked in the first half of 1985 as visiting professor at Rutgers University, New Jersey, and is currently employed at Eindhoven University of Technology.

North-Holland
Computers in Industry 6 (1985) 435–452

1. Introduction

Current information systems for production management show a remarkable paradox. On the one hand, there is a tendency to integrate more and more all kinds of stand-alone computer applications: much work is done to integrate Computer-Aided design (CAD) with Computer-Aided Manufacturing (CAM), Computer-Aided Testing (CAT) and Computer-Aided Logistics (CAL). All these integrating activities should eventually lead to Computer-Integrated Manufacturing (CIM). On the other hand, even the more conventional computer applications in the production area require databases of an overwhelming complexity; further integration will no doubt enlarge the complexity considerably. This complexity is causing many coordinatory problems in the management of data, the development of software, and the interpretation of information. Decentralization of hardware and distributed processing are often presented as a solution to the problems of complexity. However, this solution pertains to the datalogical level of design only, whereas the problems of complexity exist to a large extent at the organizational (systelogical) level and at the conceptual (infological) level.

We shall elaborate on that complexity; a more detailed treatment is given in Section 2. Conventional computer applications of production companies could for example comprise the following applications:

- Accounting (computation of standard prices, budget control, collection of information for annual reporting)
- Demand management (long term, medium term, short term forecasting, quotation of prices and delivery dates order entry etc.).
- Production/Inventory control (Master Planning, Materials Planning, Workorder Release, lot-sizing, setting safety stocks etc.).
- Manufacturing (Shop floor control, Work-order

Routing, Machine maintenance quality control, tool control).
- Product development (specification of products, options, allowable combinations of options, effectivity dates)
- Process engineering (investment in equipment, manufacturing instructions, training etc.).

We shall call these application-categories *functional task areas*; organizations that are hierarchically structured according to the functional task areas will be denoted as *functional organizations*. We conjecture that functional organizations cause many coordinatory problems in information systems development and in data management. This is due to the fact that production management goals such as due-date reliability, product-quality, flexibility and efficiency are usually related to several functional task areas. In our opinion, an organizational structure based on the concepts such as self-contained tasks and lateral coordination is more promising (cf. Galbraith [1]).

A closely related topic enters the scene at infological level of design. The current methodology for infological design distinguishes between data-modelling and process-modelling. The conceptual data-modelling techniques describe a universe of discourse that, in principle, is homogenous. In other words, these techniques provide no facilities for the decomposition of a conceptual datamodel into several parts. In our opinion such a decomposition is made possible by the concept of ownership of data. The ownership of data specifies, which organizational function has authority to change that data. This infological concept is discussed further in Section 3.

At the datalogical level, the concepts for decomposition take the form of distributed processing. However, distributed processing is nowadays considered to be a datalogical solution for a datalogical problem. This problem is formed by the poor performance of large mainframes when dealing with a large number of on-line connected users. The solution offered by distributed processing consists of the employment of decentralized hardware, connected by a network, to solve the same problem in a technically superior way. Distributed processing aims at hiding the fact that decentralized hardware is employed. As such, it is not a concept for effective decomposition of information systems but for more efficient employment of hardware-resources. Therefore, we shall not discuss distributed processing in depth in this paper. However, the above argument is explained in detail in Section 3.1.

Instead we shall present a case study of our systelogical and infological ideas, when applied to a production organization. The case study deals with a truck-manufacturing company. For ease of discussion, we made some simplifications as compared with the real situation from which the case was derived. This case study is presented in Section 4. Section 5, finally gives some conclusions. It emphasizes the main message of the paper, viz. that there is a need for decomposition concepts in the systelogical and infological phases of information systems design.

2. Decomposition at the Organizational (Systelogical) Level

2.1. Statement of the Problem

A first problem with current databases in the production area is that they become more and more complex. Not only the number of entity types and attributes is increasing, but also the heterogenity of the users. When many users are involved, sharing the same database, it becomes difficult to attain consencus on the precise meaning of many entity-types and attributes. Consider for example the bill-of-material structure in a production company (cf. Van Rijn and Wortmann [2]). A product-development team will structure a bill-of-material primarily according to product-engineering disciplines. In a truck manufacturing company, the brake-system of a truck will occur as an item in the product-engineering view (the first view). Such a brake system, however, is never assembled during actual assembly. Therefore, the manufacturing engineers of the assembly line will employ a different bill of material (the second view). The material planners, in turn, are interested in the common parts and in the optional parts of a truck (cf. Orlicky, Plossl and Wight [3]). Therefore, the material planners will employ another bill-of-material (the third view); and so on. This example illustrates, that users with different tasks in the organization have not only different views on the data in the database, but require different data with respect to the same objects encountered in reality. Differences appear not only

in subsets of attributes, but also in aggregation levels, sets of occurrences and actual relationships, etc.

An important organizational problem is concerned with the responsibility for the correctness of data. As more and more organizational functions are sharing data, it becomes nearly impossible to establish a clear organizational picture of these responsibilities. This problem emerges for example when engineering-changes require a great number of approving meetings, where nearly all functional task areas are involved. After approval, engineering changes nevertheless remain to cause problems, because the meaning of all attributes in the database and its consequences for application programs are far from transparent.

In many practical situations, these problems are dealt with by appointing the responsibility for the correctness of the data to one of the organizational functions involved. In the truck manufacturing company of our case study, for example, the product-development group was responsible for the correctness of the bill-of-material. Such an approach is not necessarily a solution to the problem. If the product development group is not aware of the interests of other users, the problem is merely deferred to a later stage.

The problem of data-correctness for many different users is not the only problem. A similar problem emerges with respect to software-development. Here too, many users are involved. More important is the fact that each new application assumes a certain semantical structure of the data. Therefore, the meaning of the data is often specified by the existing application software, and a change of this meaning is hardly possible. In the truck-manufacturing case this point emerged in the following way. A project was started to develop a system for coordination of the material flow between several plants. However, the bill-of-material was only available in the form of a detailed assembly structure, describing the way in which a truck is built within the assembly lines of the plants. This detailed information is not very suitable for inter-plant material coordination. An aggregated bill-of-material cannot be derived, if the necessity of such an aggregation has not been specified beforehand.

2.2. Analysis

In order to analyze the above mentioned organizational problems, we shall take rescue to the so-called contingency theory to organizational design. The work of Galbraith [1] is an excellent starting point. According to this work, the central question in organizational design deals with the ways in which complexity and uncertainty in executing tasks are reduced. Before proceeding, we shall shortly expose the criteria according to which organizations are structured.

The classical "departmental models" of organizational theory have studied several alternatives, while adhering to the principle of "unit of command". The first alternative stresses the efficiency of the division of labour. In this view, many different specialized resources (functional task areas) are distinguished, and the organizational structure is stratified according to the similarity of these resources. We shall call such an organizational structure input-based. This type of organization requires quite some effort to coordinate ongoing processes that share resources from several functional task areas. The other alternative to clustering subtasks is to group tasks that have a common mission. We shall call these organizational structures output-based. In output based organizations, the problem emerges of how to prevent inefficiency due to a reduction of the division of labor.

In our view, the organizational problems mentioned earlier with respect to data-management and software development, stem from an input-based task design. Because all functional task areas are organized into separate groups, decisions on engineering change, for example, go across the whole organization before they can be implemented. Approval of changes in the database and approval of software for e.g. production control cannot be delegated to subgroups. Each organizational function (viz. each functional task area) envisages much uncertainty in performing its task, because each functional task area is strongly dependent upon other functional task areas.

2.3. Outline of an Organizational Solution

When organizations face an unsufficient performance (such as a poor data-management or a slow software development) several alternatives are pos-

	ALTERNATIVES			
	CREATION OF SLACK RESOURCES	CREATION OF SELF-CONTAINED TASKS	INVESTMENTS IN VERTICAL INFORMATION SYSTEMS	CREATION OF LATERAL RELATIONS
REDUCE THE NEED FOR INFORMATION PROCESSING	X	X		
INCREASE THE CAPACITY TO PROCESS INFORMATION			X	X
CHANGE ORGANI-ZATIONAL STRUCTURE		X		X
DECREASE REQUIRED LEVEL OF PERFORMANCE	X		X	

Fig. 1. Design strategies for reducing task uncertainty (derived from Galbraith [1]).

sible (cf. Galbraith, see Fig. 1). Often, more alternatives are employed together.

The first alternative, creation of slack resources, is what usually happens if there is no active response to organizational problems. In many organizations this takes the form of time-slack: decision procedures take a lot of time, and, therefore, decrease the level of performance in this respect. This happens in the engineering change problems described in subsection 2.1.

If the organization responds actively by reducing the need for coordination (i.e. reducing the need for information processing) it may choose to create more self-contained tasks. This means, that the organizational structure is not based primarily on clustering similar skills, resources or professions ("input-based task design") but on grouping together the different skills and resources required to fulfill a certain mission, such as the production of a certain type of product (e.g. cabines, motors, gearboxes–"output-based" task design). In the production area, self-contained tasks may take the form of group-technology at shop-floor level; it may also take the form of product-divisions at corporate level. If an organization moves towards self-contained tasks, this usually means a decentralization of staff skills and a distribution of resources. This goes together with fewer distinctly different functional task area.

As depicted in Fig. 1, the organization can also choose to increase the capacity to process information. There are two ways of achieving this goal. The first is to invest in information systems for the purpose of improving the hierarchical communication lines. In order to stress this purpose, the strategy is called: the investment in vertical information systems. If such a strategy goes together with the strategy of creating self-contained tasks, these information systems are aiming at generating aggregated information. Otherwise, they are aiming at generating quickly a lot of detailed information. The other strategy for increasing the capacity to process information is the creation of lateral processes. To some extent, these two strategies are complementary.

The creation of self-contained tasks brings discretion at lower levels of the organization. A self-contained task group does not need information from other groups when solving problems. "However, if discretion is to be increased at lower levels without reducing resource sharing, lateral relations are required". (Galbraith [1], p. 47). This is especially the case in materials coordination in the production area, where one production unit or plant delivers the resources (viz. the materials) for another one. Lateral relations may take various forms. In its weekest form, a lateral relation is implemented by direct contact between the

managers involved. Somewhat stronger forms are task forces or permanent teams. If the coordinatory aspect requires substantial leadership, a linking-managerial role such as a project leader may emerge. Finally, if the coordinatory aspect is quite critical, dual authority relations are created, and a matrix-design emerges.

Returning to our problem statement the outline of our organizational solution can be summarized as follows: the most appropriate way to eliminate coordinatory problems in decision-making and, therefore, in data-management is to create self-contained tasks. This enables the design of local databases and local applications for which the self-contained task group is responsible. For the remaining coordinatory problems, a lateral relation should be created. If this lateral relation takes the form of a separate organizational entity, the responsibility for a part of the common database can be allocated to this entity. Such an organizational design allows us to allocate the responsibility for large parts of the database to separate groups. If the responsibility for parts of the database is allocated to distinct self-contained task groups, it becomes possible to design vertical information systems by formally deriving information from "local" databases. We shall clarify this approach in Section 4 for the case of the truck manufacturing company.

3. Decomposition at the Infological Level

3.1. Statement of the Problem

As stated earlier, the problem with common databases is that they become more and more complex. Not only the number of entity types in the database is increasing, but also the heterogenity of the users. With respect to modern production databases the users have different tasks in the organization and therefore different views on the data. Whether the properties of entity types and the relationships between them, are relevant for a person, depends on the task of that person. Differences appear not only in subsets of attributes, but also in aggregation levels, ways of aggregation, versions, requirements for actuality etc. It is very difficult to design a common datastructure that is non-redundant and reflects the views of all users. It is even more difficult to maintain a consistent structure in a flexible environment with frequent modifications of tasks and rapidly evoluating concepts.

The problem is comparable to the "software crisis" in programming that occurred when programs became larger and more complex. A program consists of a collection of variables (data elements) and a collection of instructions. Instructions modify variables and variables influence the effect of instructions. As long as all variables and all instructions fit into one or two pages of paper, there is no problem. When the length of a program becomes more than 10 (maybe 20) pages it is impossible to understand which variables influence what instructions and vice-versa. The cause of errors and the effect of modifications can no longer be kept under control. The solution of the problem has been found in modular programming [4]: the big program is split into modules that cannot see or influence each others variables. Each module is only a few pages long and can be well understood. The communication between modules is via explicitly stated parameter lists, or via restricted common areas. The problem with common areas is, however, that they tend to grow until they contain most of the variables of all modules, which implies the revocation of the control problem. The problem with a common database is that it is very much like a huge common area: all data elements (variables) can be modified by all application programs (modules), so it is very difficult to understand the cause of errors or to control the effect of modifications.

This problem is causing a "data administration" crisis that is comparable to the software crisis in programming: the uncontrollable impact of modifications on the database structure becomes prohibitive for organizational changes.

Another process is evolving in the other direction: it is causing an integration problem that is closely related to the decomposition problem. Many local information systems are introduced in the form of special functions with embedded microcomputers. In the office environment these are text processors, spreadsheet applications, personal databases, electronic mail systems etc. In the factory these are robots, electronic testers, transport systems, automated warehouses, CNC-machines, computerized scheduling systems, quality control systems, process control and monitoring systems etc. These functions are introduced as stand alone

applications and humans take care for the interfacing between them.

As indicated in Section 1, distributed processing is often advocated as a solution for the problem of integrating these local information systems [5]. "Distributed processing" stands for a data communication concept that automatically provides facilities for interfacing different processors. In earlier days, these interfacing problems were solved by point-to-point connections, whereas nowadays Local Area Networks are rapidly entering the scene. The connections are standardized according to the Open Systems Interconnection (OSI) architecture. However, these concepts provide nothing more than a connection at the syntactical level. Therefore, they constitute a major step at the datalogical level of design but at the same time, they provide no solution at all at the infological level of design.

The problem is like that of a person who wants to make a phonecall in a foreign country. If he has found a telephone, knows how to operate it and knows the right telephone number, there can be a connection, but there will not be any communication if he does not speak the language. Even if that person and the one he wants to communicate with speak a common language, there cannot be communication unless they have a common frame of reference. The computerized functions that are introduced in the organization originate from different suppliers and are programmed by different people, so these computers speak different languages. If they are to communicate, even via the most perfectly standardized network, they have to learn a common language. This implies that software modifications have to be made, most times in both partners. Humans can have misunderstandings due to slight differences in their frames of reference, but in most cases they are able to correct them in time because their communication is syntactically and semantically redundant and because humans are very adaptive. Computer communication is not redundant, at least not semantically and computers are not adaptive. Because of that, the slightest difference in frame of reference between communicating computers can be disastrous.

Especially in production control there is still very little standardization in concepts, so it cannot be expected that computers, programmed by different people with different backgrounds, will have compatible frames of reference. This makes the CAD/CAM integration such a heavy task. It is difficult to integrate a number of computers, with different functions from different suppliers, into a reliable distributed system. Complexity increases with an order of magnitude if components in such a system have to be exchangeable, for instance to be able to adapt the system to changing requirements, new generations of equipment or the need to switch to other suppliers of equipment.

Distributing programs and data will not resolve the "data administration" crisis. Interconnecting distributed computers will not result in an integrated system. Structured decomposition and flexible integration, in human organizations as well as in computerized systems, mean that each function in the system has distinct and well defined tasks, responsibilities, data and frames of reference. It also means that the functions are interfaced by well-defined coordination procedures, common data and consistent overlap in frames of references.

3.2. Analysis

In the database world, a frame of reference or universe of discours is modelled with a so called conceptual data model [6]. Conceptual data models have been developed to support the design of common databases. Current models however have no concept for decomposition. The concept of subschemas or external views is able to express the fact that not every body needs to see all data elements, but it is not sufficient to model decomposition. The idea of decomposition is that the corporate task is decomposed into subtasks, so that each subtask is highly independent from others. Subtasks should have a minimal need for coordination and communication with other tasks and should not overlap with each other. The problem with subschemas is that they must overlap, to allow communication between tasks, so they are not an appropriate concept for decomposition.

Organizations have to be decomposed in order to stay flexible and manageable; the conceptual data model of an organization has to be decomposed for the same reason. This means that it has to be split in non-overlapping modules, as much as possible corresponding to the self-contained tasks. Because the self-contained parts of an organization have to communicate in order to coordinate

their actions, there must be an overlap of concepts between the modules of the conceptual model. By controlling the consistency of the overlapping concepts it becomes possible to manage the coherence of the system. Modifications in data structures can be admitted freely as long as they do not overlap with the structures of other users, but require organized control when they affect overlapping structures. There is a need for formal techniques to check the consistency of overlapping concepts.

3.3. Outline of a Solution

In our opinion, the decomposition of conceptual data models requires that the notion of ownership of data is introduced in the conceptual model, in order to link entity types with organizational functions. This notion can be illustrated with the distinction between the original and the copies of data. It is clear that there can exist only one original of any datum. When an original datum is changed, all holders of copies of that datum have to be provided timely with a fresh copy. It is useless to modify copies, because the modification will be lost as soon as a fresh copy originates. The holder of the original of a datum is the organizational task that is responsible for the contents of that datum. Responsibility for a datum implies that it requires a decision to modify the datum, regardless whether the datum represents some observable fact or some human decision. It is to be preferred that for each type of decision there is only one responsible organizational task. In cases where a decision requires the approval of different organizational tasks, there has to be some committee that can be viewed as a separate organizational task, or the decision can be viewed as the conjunction of a number of independent decisions. Situations where different tasks are responsible for the same type of decision should be very rare.

Accordingly, in well designed organizations it should be possible to point to one single organizational task as the responsible task for the contents of any datum. The situation where different tasks are to be allowed to modify the same datum occurs mainly where different tasks share a single resource. In most cases this situation will require a separate organizational task to coordinate the allocation of that resource. Since the notion of ownership implies that every object has one single owner, be it a person or a corporation, we will call an organizational task that is responsible for the contents of a datum the owner of that datum. Other organizational tasks, that hold copies of that datum we will call readers of the datum. In order to be able to decompose a conceptual schema, we propose to model entity types so that all data elements, belonging to one entity type, have the same owner. As a consequence of this, conceptual datamodules can be formed of entity types that are owned by the same organizational task. As long as ownership is unambiguous, these datamodules will not overlap.

Now every datamodule will consist of a part that is read only by the owner and a part that is read by one or more other organizational tasks. The first part can be referred to as the private data of the owner, the second as the public data. Every organizational task is authorized to alter only the contents of its own (public or private) data. Every organizational task is free to alter the structure of its private data. Modification of the structure of public data requires some organized coordination.

A number of different semantic datamodels is available [7,8]. They have some things in common:

- they are designed to model the semantics of data without referring to the technique of representation,
- they model a frame of reference as a schema that represents the total set of possible realities within that frame of reference,
- an element of that schema, representing a possible reality, is called an occurrence of the schema.

The most influential model is the relational model [9], which has the drawback that it is rather poor in expressing semantics. The set model (cf. Remmen [10]) is closely related to the relational model, but more precise in its definitions and therefore much more powerful in expressing semantics [11]. Without going into the details of definitions and specification languages we will describe the use of a datamodel, basing ourselves upon the concepts of the relational and the set model. For a more detailed introduction to the interpretation of a datamodel we refer to the appendix (A1). Here we will give a quick introduction to a simple example. Fig. 2 shows an example conceptual schema in the form of an extended Bachman diagram [12].

The schema contains the entity types ITEM, BOM, ORDER and COMPLETION. The arrows between entity types represent 1:N relationships.

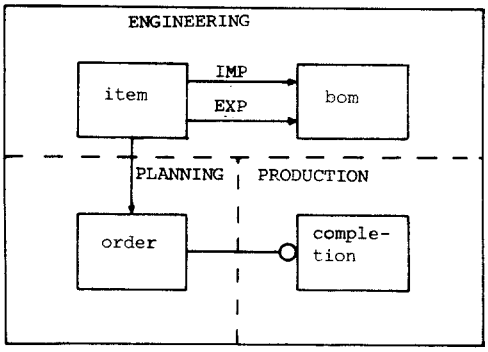

Fig. 2. Example of a conceptual schema diagram.

The arrow from ITEM to ORDER indicates that every ITEM-occurrence can have zero or more ORDER occurrences and that every ORDER-occurrence refers to precisely one ITEM occurrence. The two arrows between ITEM and BOM represent the bill of material in explosion resp. implosion direction. Every BOM-occurrence refers to precisely one composite ITEM (imp relationship) and precisely one component ITEM (exp relationship). The arc between ORDER and COMPLETION (- - - - - 0) is an extention to the Bachman convention and depicts a generalization relationship between ORDER and COMPLETION. It expresses that an ORDER can be completed or not: an ORDER occurrence can have zero or one COMPLETION-occurrences and every COMPLETION-occurrence refers to precisely one order-occurrence. This relationship is called a generalization (cf. Smith and Smith [13]) because ORDER is the generalization of "completed orders" and "uncompleted orders". Completed orders have properties (completion date) that uncompleted orders have not. The dotted lines are another extension to the Bachman convention. They devide the schema into modules that are owned by the indicated organizational tasks (engineering, planning and production). This indicates that these organizational tasks are responsible for, and decide about the contents of the entity types.

This example shows how the concept of ownership of data provides a criterium for the decomposition of a conceptual datamodel. Thereby it links the entity types to organizational tasks and shows how responsibilities are structured. When models like this are made for each of a set of related organizational tasks, it will be possible to distinguish between private and public data elements and to check whether the different views upon the structure and the ownership of the data are consistent. It should be noted that structures as viewed by readers need not be identical to, but must be derivable from those as viewed by the owner. This will be illustrated in the next Section.

4. Application to a Case: A Truck Manufacturing Company

4.1. Introduction

The approach outlined in the previous Sections is now applied to the case of a truck manufacturing company. For ease of discussion, the case has been simplified, without changing the essentials. The production division of a truck manufacturer in The Netherlands consists of several plants. The trucks are assembled in a final assembly plant. The major components of a truck, viz. the motor, the cabine, and the axles are assembled in sub-assembly plants. The materials for these assembly operations are either provided by external suppliers or by a component manufacturer plant. There are two component manufacturing plants, viz. the pressing plant and the machining plant. The material flow is depicted in Fig. 3.

The existing production organization is depicted in Fig. 4. For each functional task area, there is a central department. These functional task areas are:

a. production/inventory control (P/C)
b. manufacturing instructions and routing (MIR)
c. quality control (QC)
d. product development (PD)
e. purchasing (PUR)
f. machine maintenance (MTN)
g. financial control (FIN)
h. personnel (PER).

Within the six plants knowledge with respect to the various functional task areas is quite limited. Hardly any authority is delegated, except for the authority to take decisions according to well known procedures and rules. These procedures and rules are established centrally, and all research with respect to a functional task area is also performed centrally.

The information systems (IS) department is centralized for the company as a whole. It is quite natural that the information systems department

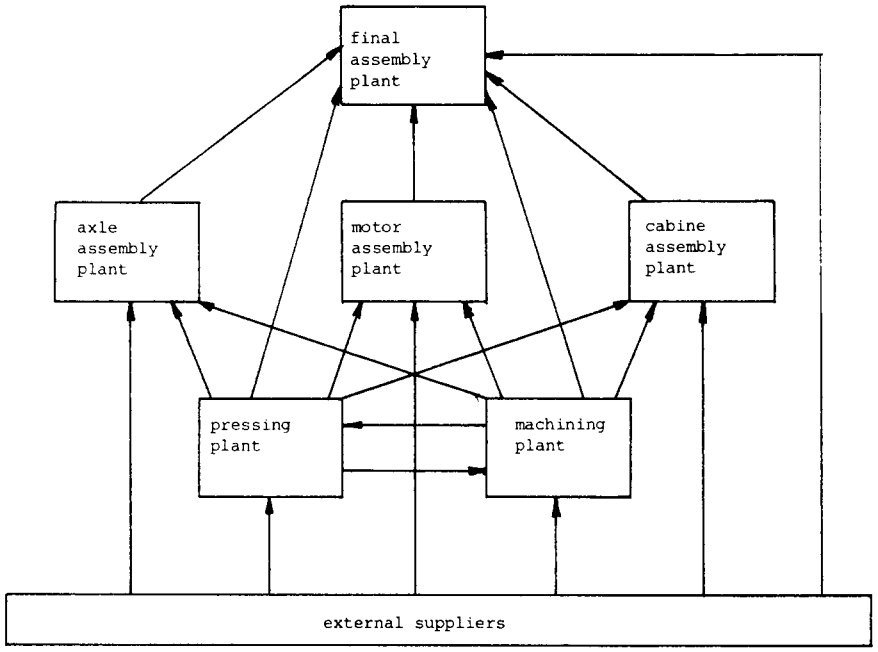

Fig. 3. Material flow for the truck manufacturing company.

has hardly any contacts with the plants, because all knowledge is available only in the central departments. Therefore, the central departments provide information systems analysts and project-leaders who are responsible for information systems development in various functional areas. This

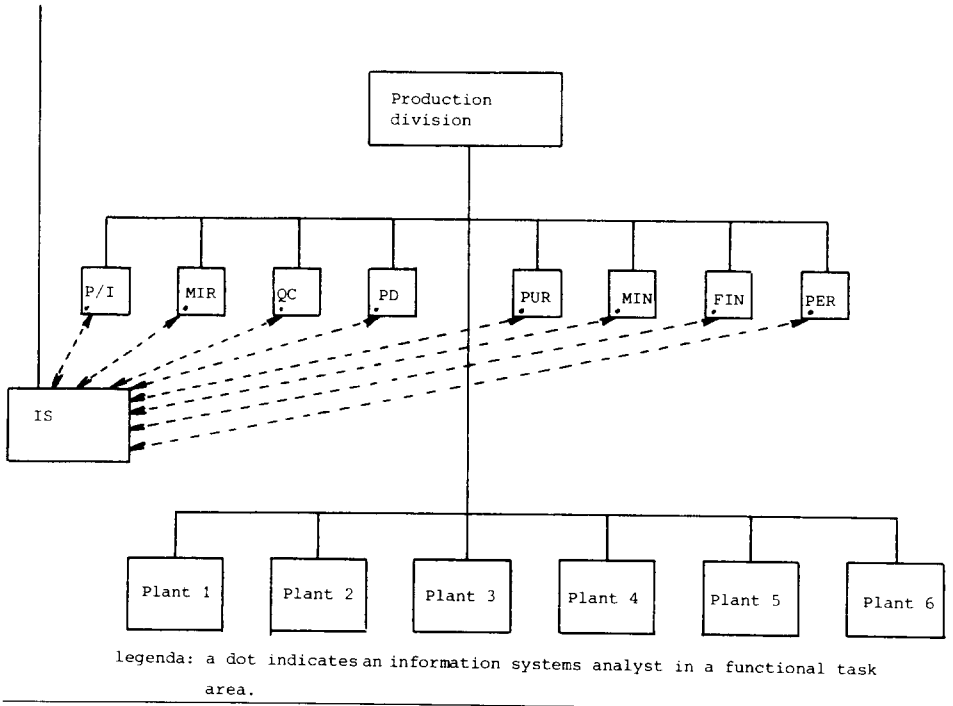

Fig. 4. Existing organization.

situation was increasingly unsatisfactory, because the applications of several functional task areas became more and more interdependent. For example, a new system for coordinating the material flow between plants (within the production/inventory control area) could not be realized without detailed involvement of specialists from product-development, manufacturing instructions and routing, purchasing, and other functional task areas.

4.2. Systelogical Redesign

Facing this situation, the management of the production division started a strategy for decomposition of the information system. This strategy started by making the plants more self-contained. To pursue this strategy, the central departments had to delegate their authority for systems development partly to the plants. At the same time, knowledge with respect to several functional task areas within the plants had to be increased considerably. This required training and management development plans within the plants.

The delegation of authority may differ for different functional task areas. The task areas "Manufacturing Instructions and Routing", "Machine Maintenance", and "Quality Control" could be delegated to a large extent. The responsibility for systems development for "Purchasing", "Product Engineering", "Personnel" and "Finance" remained with central departments. The solution for "Production/Inventory Control" is somewhat in between, as we shall explain in detail now.

4.3. Levels of Control in the Production/Inventory Area

Three levels of control are distinguished for our present purpose, viz.:

1. Master Planning, for the coordination of production levels and materials between plants
2. Plant Materials Coordination, for the planning of assembly lines and departments within each plant
3. Shop Floor Control, for the control of workorders within departments or within assembly lines.

Considerable discussion preceded the establishment of the Master Planning function. If this function does not exist, the plants are much more autonomous with respect to Production/Inventory control. However, the well-known advantages of integral material flow control lead to the decision to keep a (central) Master Planning function. Especially a flexible response to changing requirements from the Sales department establishes the need to perform the planning of critical materials and the coordination of the volume of output centrally. A centralized Master Planning function enables a quick response to changing customer demands, because all plants can be consulted in parallel to each other (as opposed to a sequential information/decision process, which would take much more time).

The Master Planning function starts from a delivery pattern requested by the Marketing-and-Sales division. This so-called Sales Master Delivery Schedule is translated into a Master Delivery Schedule (MDS) for each plant, through offsetting by slightly exaggerated manufacturing leadtimes. This slight exaggeration enables the Plant Material Control function to perform some internal scheduling without disturbing the MDS of the supplying plants. In this way, the "self-containedness" of the plants is enhanced. Of course, these MDS-patterns of plants are only specified for items considered to be critical. These items are motors, axles and cabines, and several items delivered by external suppliers. For the pressing plant and the machining plant, the control of the volume of production is more important. This requires explicit Input/Output Control, in terms of aggregated capacity. The Master Planning function requires a bill of material that does not show items moving within plants, but only critical items moving between plants. Such items are called: Master-planning items (MP-items). Furthermore, some global capacity constraints are to be satisfied in creating Master Delivery Schedules, viz. certain mix-constraints in the assembly plants and volume-oriented input/output constraints in the component manufacturing plants.

The Plant Materials Coordination starts from a Master Delivery Schedule specified by the Master Planning function. Based on the constraints, set by this MDS, the Plant Materials Coordination creates plans for workorders for several items, resulting in a plan for realization of the required Master Planning-items; this plan is called: the Master Production Schedule (MPS). The difference between the

MPS and the MDS represents a planned inventory of finished items. In order to perform its functions, the Plant Materials Coordination has a detailed bill-of-material at disposal, created by Product Development. Each item in this bill-of-material is manufactured by a production unit: such a production unit is a department of a component manufacturing plant, or an assembly line in an assembly plant. The fact, that each item is manufactured by a production unit, requires considerable coordinatory effort by the (still centralized) Product Development department; the definition of "items" requires approval from (some) plant management!

Another interesting point deals with the control of changes, requested by the Marketing and Sales department. As stated above, such requests are treated by the Master Planning decision function. However, when a certain truck is already in process in the cabine-plant, a change in the cabine is no longer allowed. In order to behave correctly in such situations, the following procedure is established. Shortly before the actual release of an order to a plant, the corresponding customer-order is firmed up. This firming-up consists of the attachment of a customer-order to a manufacturing order in each of the assembly plants. If an order has been firmed-up, the plants are allowed to release the order (after allocation of materials). As soon as an order is firmed up, the Master Planning function is no longer allowed to make any changes to the order (as far as the plant in question is concerned). If the order is firmed up, but not yet released, the Master Planning function is only able to start negociations about an order, but the decision-making power lies within the plant. For this reason, the Master Planning function has to know, whether orders are released.

The third level of control is the shop floor control. At this level, the flow of workorders through the production units is managed. In performing this task, the shop floor control function dispatches work-orders to man-machine facilities; in order to perform this function, the operations of the work-order have to be known. Furthermore, shop floor control is controlling the issue of materials for operations. The primary goal of shop floor control is to maintain due-date reliability of work-orders (while maintaining the production rate of the production unit as a whole – cf. Bertrand and Wortmann [14]).

4.4. Infological Design for Production / Inventory Management

The infological design is depicted in Figs. 5, 6 and 7 and in the corresponding Tables in Appendix A3, A4 and A5. These figures show the conceptual data-model for Master Planning, for Plant Material Control and for Shop Floor Control. Each data model is primarily a "readership" datamodel: it specifies the data, available for each function. However, the "ownership" of data is indicated in the drawings as well. Let us now consider the figures in more detail.

The Master Planning data model is depicted in Fig. 5. The top of Fig. 5 shows that there are several plants defined (under authority of the general management); with each plant, a number of global capacity constraints are associated. Furthermore, a number of master planning items and a master-planning bill of material structure are defined by Product Development. The MP-items are related to the global capacity constraints by capacity requirements (to be specified by Manufacturing Instructions and Routing).

The Master Planning function adds information to this (static) picture, as shown in the middle of Fig. 5. A number of future long-term periods are distinguished, and a series of MDS-plans is created for each MP-item. Furthermore, the firming-up of MDS-orders belongs to the responsibility of the Master Planning function. Fig. 5 shows furthermore, that the release of orders and the authority to consider an order as finished rests with the Plant Materials Coordination.

For each plant, the data model of the Plant Materials Coordination function is depicted in Fig. 6. The bottom of Fig. 6 shows entities and relationships that are created under the responsibility of the Plant Materials Coordination function. The top of Fig. 6 shows data, that are only readable for the Plant Materials Coordination. We shall shortly discuss this Figure. The plant management of each plant establishes the Production Units to be distinguished, and a number of critical capacity constraints per Production Unit. Plant Manufacturing Instructions and Routing specifies the amount of each capacity constraint, required by each plant-item. The items, in tern, are specified by Product Development. Some items are MP-items, and for these items is always an MDS specified by the Master Planning function.

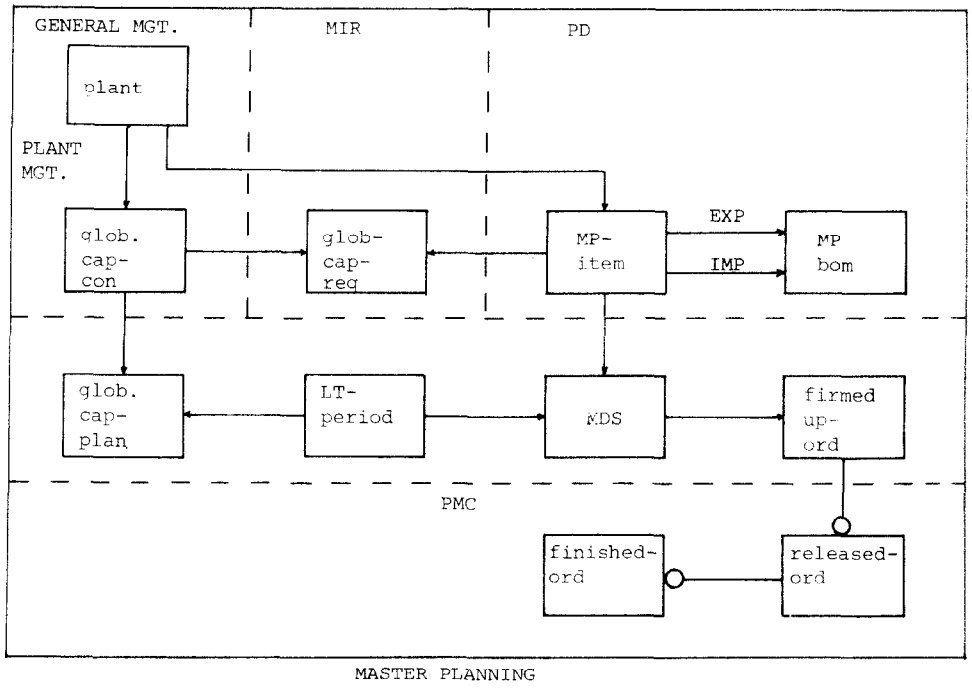

Fig. 5. Conceptual schema of the Master Planning Function.

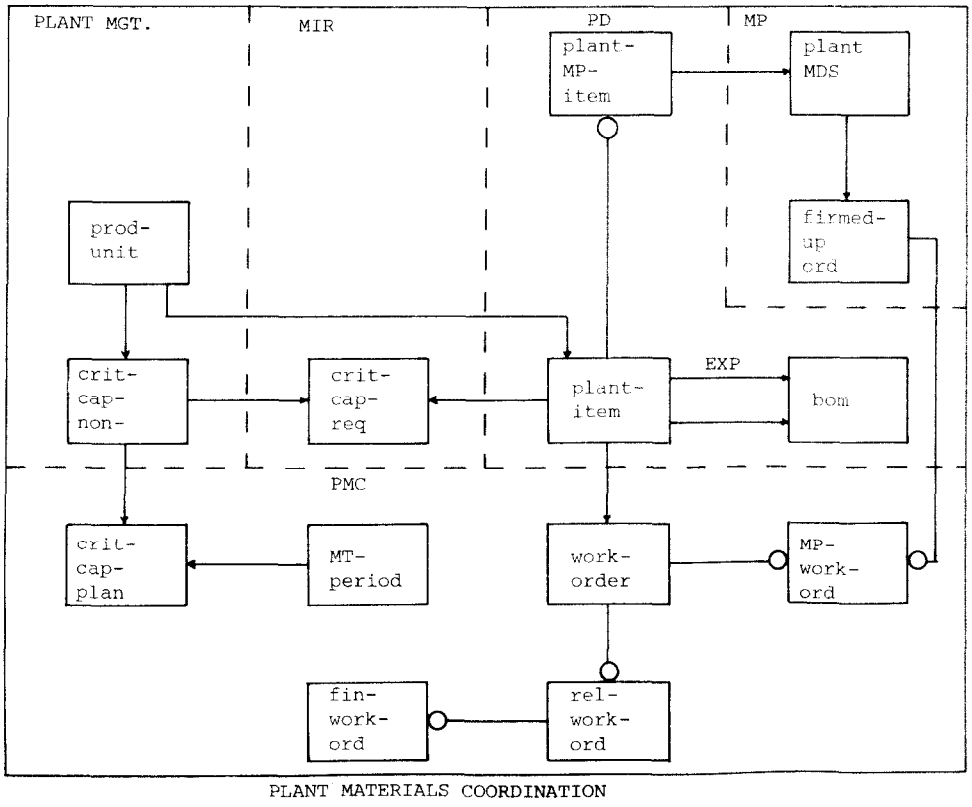

Fig. 6. Conceptual schema of the Plant Materials Coordination Function.

The Plant Materials Coordination function creates a plan of work-orders for MP-items (the so-called Plant Master Production Schedule). This plant-MPS should always produce the orders earlier than the requirements of the plant-MDS. Based on this, his MPS work-orders are planned for other plant-items. These plans can be translated into capacity requirements of critical capacity constraints per Production Unit, yielding capacity loading plans over a number of medium-term periods. The latter plans are also created under the responsibility of Plant Materials Coordination. Finally, the release of work-orders to Production Units and the authority to consider a work-order as finished rests with the Plant Materials Management, as mentioned above.

For each Production Unit, the data model is presented in Fig. 7. This Figure shows that each Production Unit consists of a number of facilities. For each plant-item, the standard operations and routing are specified by the Plant Manufacturing Instructions and Routing function. The required materials for each operation are also specified here. The management of the Production Unit is responsible for the flow of work-orders through the Production Unit. Therefore, the dispatching of shop-operations and material issues is the area of responsibility of the Production Unit management.

A comparison of Figs. 5, 6 and 7 illustrates how the modular data model shows the relevant concepts for each of the three control levels. The ownership of the entities, models how the responsibilities are allocated. The differences between the structures of Figs. 5, 6 and 7 show to what extend the respective control functions have different views on the relevant concepts. With respect to the concept of planned quantities for instance, Fig. 5 shows that, at the Master Planning level, MDS links MP-item to LT-period: the Master Delivery Schedule defines for each MP-item the quantity to be produced in each LT-period (see also the attribute table in Appendix A3). In Fig. 6 we see that, from the Materials Coordination's point of view, a work-order is not directly linked to a MT-period, but to a specific due date (see attribute table in

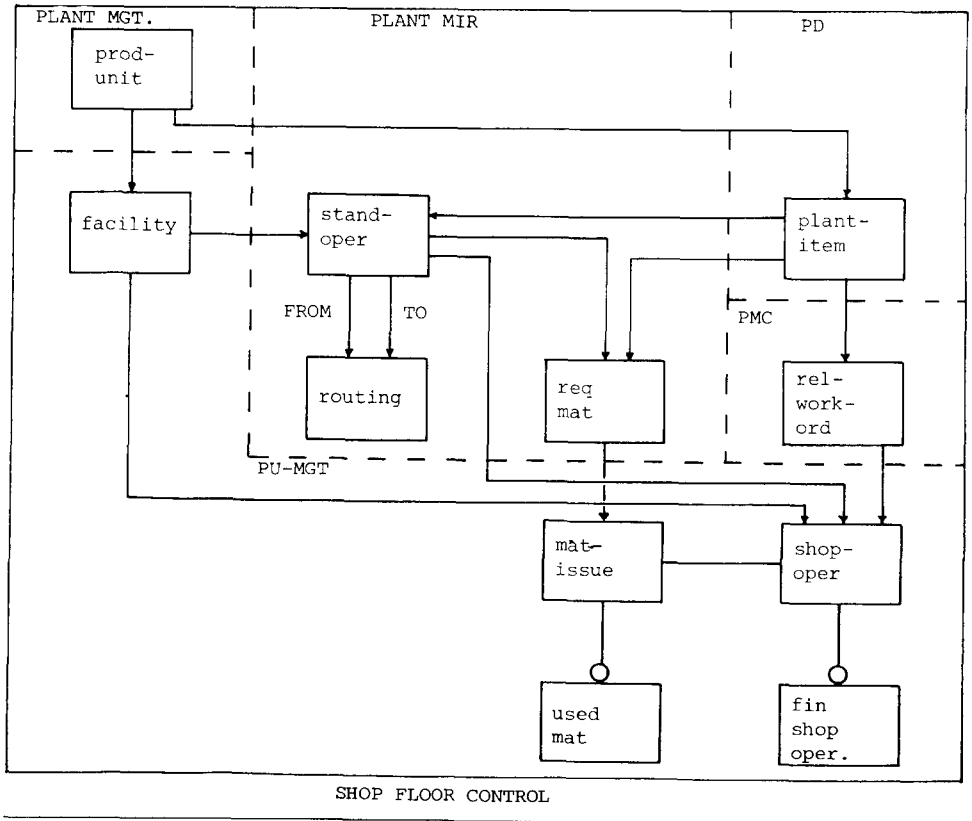

Fig. 7. Conceptual schema of the Shop Floor Control function.

Appendix A4). Fig. 7 shows that the Shop floor Control function is interested in only the released work orders.

5. Conclusion

In this paper we showed, that there is a need for an organizational and infological approach to the decomposition of information systems. From an organizational point of view, there is quite some theory available for the contingency-approach to organizational design. However, more experience and research is required for the application of these ideas in relation to the redesign of information systems. We described a (simplified) example of a case in-practice.

From an infological point of view, data-modelling methodology is a logical starting point. However, this methodology has to be extended, in our opinion, by the concept of "ownership". We indicated for our case study, what such an "ownership" concept would yield. However, a long way has to be gone before such a concept is included more formally in data-modelling techniques.

Acknowledgements

The ideas presented in this paper were developed in close cooperation with Karel van Eynde and Joost Timmer from DAF-Trucks Manufacturing Company Eindhoven, The Netherlands. We owe much insight in data-modelling techniques to Frans Remmen, Mathematics and Information Science Department, Eindhoven University of Technology.

References

[1] Galbraith, J.: "Designing Complex Organizations". Addison-Wesley, Reading Mass., 1973.

[2] Rijn, Th.M.J. van, and J.C. Wortmann: "Structuring of Products and their Descriptions", working paper. Eindhoven University of Technology, 1985.

[3] Orlicky, J.A., G.W. Plossl and O.W. Wight: "Structuring the Bill of Material for MRP". Producion & Inventory Management, 4th qtr 1972.

[4] Bates, J. (ed.): "Structured Programming. Infotech State of the Art Report". Infotech International, 1976.

[5] Martin, J.: "Distributed Data Processing: the Opportunity and the Challenge". Savant Research Studies, Savant Institute, 1979.

[6] ANSI/X3/SPARC: "Study Group on Database Management Systems: Interim Report 75-02-08". In: ACM SIGMOD Newsletter, FDT, Vol 1, No. 1, 1969.

[7] Tsichritzis, D.C. and F.H. Lochovsky: "Data Models". Prentice-Hall, Englewood-Cliffs, N.J. 1982.

[8] Brodie, M.L., J. Mylopoulos and J.W. Schmidt (ed.): A Conceptual Modelling". Springer Verlag, New York, 1984.

[9] Codd, E.F.: "A Relational Model for Large Shared Data Banks". Comm. ACM 13, pp. 377–387, 1970.

[10] Remmen, F.: "Databases. Grondslagen voor het Logisch Ontwerp". Academic Service, 1982.

[11] Pels, H.J.: "Vergelijking van Gegevensmodellen". Proc. NGI-SION voorjaarsconferentie april 1985, (to be published) NGI, Amsterdam, 1985.

[12] Bachman, C.W.: "Data Structure Diagrams". Data Base (New York) Vol. 1, No. 1, 1969.

[13] Smith, J.M. and D.C.P. Smith: "Database Abstractions: Aggregation and Generalization". ACM Transactions on Database Systems, Vol. 2, No. 2, pp. 105–133, June 1977.

[14] Bertrand, J.W.M. and J.C. Wortmann: "Production Control and Information Systems for Component Manufacturing Shops". Elsevier, Amsterdam, 1982.