Comparison of information systems for engineer-to-order and make-to-stock situations

Hans Wortmann

Eindhoven University of Technology, Graduate School of Industrial Engineering and Management Science P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands

Abstract

This paper investigates the nature of information systems for the support of production management. It provides a comparison between these information systems in two cases: (i) the case of make-to-stock production; and (ii) the case of engineer-to-order production.

Databases with basic product and process information have quite a different nature for these two cases. For make-to-stock production, such as classical MRP systems, the basic information should be complete, consistent and up-to-date. This information is used by planners to generate materials and capacity plans in an automated way. For engineer-to-order production, it should be allowed that basic information is incomplete, partly inconsistent, or not up-to-date. This is due to the fact that these data are used by engineers as reference data for the creation of customer-specific solutions.

Keywords: Logistics; CIM

1. Introduction

This paper will discuss information systems design for production/inventory control in customer-driven manufacturing. In order to simplify the discussion, we will concentrate on the difference between the make-to-stock situation (in Section 2) and the engineer-to-order situation (in Section 3). For both situations, we will present information systems architectures and elementary data structures. The data structures will be used to highlight differences. Other types of production systems are mentioned in Section 4, together with some conclusions.

The general architecture of software for the support of production/inventory control can be explained by Fig. 1. This figure shows four concentric circles of software. The meaning of these circles is that an outer circle relies on the existence of an inner circle before it can be programmed or installed.

1.1. Systems software

The innermost circle contains the application-independent software packages which should be available before application programs can run: "systems software". This circle contains at least the operating system of the hardware involved. However, many more application-independent software packages are nowadays available for the development of application software, such as:

- A database management system (DBMS). This is a software system which hides the details about
storage of data on physical media. In this way, the application programs only have access to data by means of a standardized interface.

- An input/output monitor (I/O monitor). This is a piece of software which hides the physical characteristics of input/output media, (generating transactions and receiving messages) for the application programmer.

- A query language. This is an easy-to-use language enabling end-users to draw some information from the database. A query language is often a part of the DBMS itself.

- An application generator. This is a software package which helps the application programmer to develop software by support of screen editing, file definition, error checking routines, etc. Usually, these generators provide support in designing transaction processing applications.

All these software systems constitute the innermost circle of Fig. 1. The application programmer will call these systems his "programming environment".

1.2. State-independent transaction processing systems

The second layer in Fig. 1 constitutes the state-independent transaction processing systems. The term "state-independent" requires some elaboration: "state" refers to the flow of orders and the flow of materials and other resources. Therefore, the "state-independent transaction processing systems" constitute all kinds of recording of products, technology and capacities which are indirectly supportive to the recording and planning of orders and materials: state-independent products, goes-into relations between products (bills-of-material), routings, capacity types, standard lead-times, and so on.

1.3. State-dependent transaction processing systems

The third layer of Fig. 1, obviously, is the state-dependent transaction processing. In this layer application software is located which monitors actual or planned state transitions of materials, other resources, and orders. Materials are received, inspected, stored, issued, consumed for assembly and so on. Other resources, such as equipment, tooling, or fixtures, are characterized by time-phased availability. Customer orders start as prospects and are transformed into confirmed orders, completely specified orders, shipped orders, invoiced orders and finished orders. Internal work orders and orders issued to suppliers and subcontractors have a similar life cycle. This layer also contains software to store and retrieve forecasting and planning data. This fact is surprising at first sight, because these data will often be generated by decision support systems which are located in the next layer. However, forecasting and planning data have to be maintained in a database for later reference, regardless of their source. Whether these numbers are computed by a mathematical model or entered manually is irrelevant.

1.4. Decision-support systems and structured decision systems

The fourth layer in Fig. 1, finally, is the layer of decision-support systems, which should aid a human decision maker. Furthermore, this layer contains the structured decision systems, which are able to perform a decision making process without human intervention (but usually on behalf the human decision maker, for example partial shipment requests belonging to a blanket order). This fourth layer is the focus...
of interest of production/inventory control theory. The present paper will put more emphasis on the second and third layers, because these are regarded to be essential for the fourth one.

During the design of information systems, the general approach proceeds from the boundary to the heart, in terms of the circles of Fig. 1. It should be noticed that the introduction of information systems in organizations proceeds from the heart to the boundary. First of all, state-independent data have to be loaded, and state-independent transaction processing systems have to be introduced, followed by state-dependent transaction processing systems. Finally, decision-support systems can be employed, which have to be fed by state-dependent data on orders, materials and other resources.

2. Information systems for make-to-stock production

2.1. General remarks

For reasons of simplicity, we will base the discussion of information systems for make-to-stock production on standard software packages for Manufacturing Resources Planning (MRP II). This can be criticized of course, because there are several other approaches. However, the conclusions which we draw for data structures remains valid for several other approaches, such as OPT, Period Batch Control etc.

The architecture of MRP II standard software packages is depicted in Fig. 2. Note that this figure is split up into two areas: the upper part represents the logistical control area, whereas the lower part represents the production activity control (PAC) area. The implementation process of a standard software package requires that a considerable number of parameters are initialized. Examples of such parameters are e.g.:

- default values for many attributes;
- unit-of-measure convention (e.g. are lead times expressed in calendar days or working days?);
- scheduling parameters.

It could be argued that these parameter data and the corresponding transaction processing software belong to a small layer between layer 1 and layer 2.

Another typical set of data which is often overlooked, is the calendar file (CAL).

2.2. State-independent transaction processing systems

State-independent transaction processing systems in MRP II software packages are first of all dealing with product information. Usually, companies have a file at their disposal in which all standard items are represented by one record. State-independent attributes of such items are numerous, e.g.

- identification number and description;
- classification codes for value, commodity type, geometry, material, primary storage location, unit of measure, ordering policy, safety stock, safety lead time, stocking policy, etc.;
- standard lead time (SLT);
- references to prime vendor, prime buyer, prime responsible planner;
- various cost attributes.

It should be noted that a number of these attributes are mandatory if certain application programs are operational that require some value for these attributes. For example, if a purchasing module is operational, many systems require that all items should be classified either as a purchased item or as a non-purchased item, the prime purchaser and the commodity code should be known, and so on. Of course, certain software packages would provide de-
fault values for mandatory attributes, but the general experience with MRP packages is that full usage of all possibilities in available applications will bring about quite some data-entry effort when new products are defined.

Finally, the point to bear in mind is that all items defined in MRP II packages are standard items.

**Bills-of-material (BOM)**

A bill-of-material for a parent item is a list of components required for this parent item. More specifically, a bill-of-material represents a set of parent–component relationships, where each such relationship is an entity on itself with at least the following attributes:

- parent identification number (key attribute);
- component identification number (key attribute);
- effectivity dates;
- quantity of component per unit of parent;
- yield/scrap factor;
- lead-time adjustment.

Several short comments are appropriate here to give the reader some impression of the bill-of-material processor usually encountered in MRP II packages.

Firstly, a component item which occurs in a bill-of-material of some parent may also act as a parent itself. In this way a multi-level bill-of-material structure may be defined. However, an item is never allowed to become (indirectly) a component of itself. In other words, if the multi-level bill-of-material structure is seen as a directed graph, this graph should be cycle-free. The bill-of-material processor should check this fact at each change of parent–component relationship.

Secondly, a component item may occur as a component of several parents. This fact is sometimes called the “modularity” (German: “Baukasten-Prinzip”) of the item structure, although it would perhaps be preferable to reserve the term modularity for less obvious properties of the product structure. The list of parent–component relationships which are defined for a specific component items is called a “where-used list”.

Thirdly, the effectivity-date attributes raise an important topic in state-independent transaction processing, viz. the fact that product structures suffer from so-called engineering changes. An engineering change is a change in the way a product is designed, manufactured or ordered. It is not uncommon for standard products to have a yearly change rate in the same order of magnitude as the number of parent–component relationships themselves. Some systems support this flow of engineering changes by including an entity type “engineering change” in the database. We will ignore this issue in the remainder of this paper.

**Routings**

A routing of an item is an ordered list of normative operations required for the manufacturing of this item out of its components. A normative operation consists of a number of manufacturing steps; their description is usually not formally represented in the database which supports an MRP package. Therefore, a normative operation is an entity with at least the following attributes:

- item identification number (key attribute);
- routing sequence number (key attribute);
- capacity unit: this denotes the (machine) capacity where the operation should take place;
- normative amount of capacity required for the operation (sometimes called operation time); this amount is usually computed from more detailed (normative) attributes, such as set-up time, run time per piece, time required to fetch and inspect tooling etc.;
- transportation time allowance; this refers to transportation of a batch from one operation to another;
- waiting time allowance; this attribute is used in scheduling the operations of a routing in order to account for queuing of a batch at a capacity unit.

From these attributes, the scheduling routine for shop floor control (production activity control) is able to compute a lead time for a work order for a specific item with a certain lot size. This lead time is the sum of transportation time allowances, waiting time allowances and computed operation items, for all operations of the routing for the item involved. This computed lead time should of course be equal to the standard lead time (SLT) attribute of the items. (However, only few MRP II packages maintain this equality.)

A routing is a precise description of the flow of a work order through the factory. A routing is used to
evaluate the impact of planned orders generated by MRP I software on detailed capacity load profiles.

2.3. Data structure diagram

In order to make the above discussion more specific from an information systems point of view, we concentrate on data structures. Bertrand and Wortmann [1] argue that data structures constitute the heart of an information system. The pictorial notation used here is derived from Martin [2].

The data structure diagram for the state-independent part of MRP packages (in their most elementary form) is given in Fig. 3. This figure shows at the top right the entity of type ITEM. For manufacturing of an item, a number of operations and a number of components are required.

The N-to-1 relationship from NORMATIVE OPERATION to ITEM represents the routing of an item. The N-to-1 relationship from NORMATIVE OPERATION to CAPACITY UNIT specifies that each normative operation should always be related to precisely one item and precisely one capacity unit.

The entity type GOZINTO-RELATION represents parents-component ("goes-into") associations in the bill-of-material structure. Each "goes-into" association is of course associated with two items: one parent item and one component item. Therefore, two relationships have to be defined from the entity type PARENT-COMPONENT to the entity type ITEM. These relationships are called "explosion" and "implosion". Via the "explosion" relationship, each (parent) item is related to zero, one or more incoming "goes-into" association. Via the "implosion" relationship, each (component) item is related to zero, one or more outgoing "goes-into" association. As mentioned, each arc (parent component occurrence) has precisely one source (component) and one sink (parent).

A software module which enables the end-user to maintain the information described in Fig. 3 is called a bill-of-material processor. Such a module provides the end-user with a number of screens to perform queries and updates, but it also enforces adherence to a number of constraints which were not explicitly mentioned above. One of these constraints states that if the bill-of-material structure is viewed as a directed graph, this graph should be cycle-free.

2.4. State-dependent transaction processing systems

The third layer in Figs. 1 and 2 is concerned with state-dependent transaction processing systems. In production/inventory control, this means recording of orders and materials, and other resources. We shall not discuss other resources here. In MRP packages, orders can take the form of customer orders and forecasts, MRP orders, (firm) planned orders for MRP items, and released orders. Materials can be on hand in inventories, they can be in-transit, or on the shop floor, as floor-stock or work-in-process. If material is being consumed, it is always for released orders. Therefore, the flow of orders and the flow of material meet each other in the released orders. In this subsection we shall first describe the flow of materials, and afterwards the flow of orders. The subsection closes with an extended data-structure diagram.

Within the MRP packages, the flow of material is always recorded in terms of items (part numbers). An item uniquely identifies materials, and two material occurrences with the same item number are exchangeable from a logistics point of view ¹.
The fact that all physical materials are identified in MRP systems by part numbers has important consequences. For example, it is difficult to represent the fact that different parent items result from one work order in MRP systems. Such a situation occurs in Group Technology, where different parts from a common family are processed together in one batch. A similar situation occurs in industries, where sorting of different products from the same lot leads to several parent items being produced in one work order. Still another example is presented by "byproducts" which often inevitably appear in process industries. All these situations are not easily treated in MRP II software packages. They require (considerable) extensions, which have been shortly discussed by Wight and Landvater [3].

All shipments and receiving transactions with respect to physical material should be divided into planned and unplanned material movement. This is required for later summary reports on inventory transactions. More important, however, is the fact that a planned material movement requires two transactions: one for the material and one for the order. For example, if a material is received according to a scheduled receipt, the quantity on hand is increased and the scheduled receipt is deleted (from the logistics point of view). If material is issued for a certain (released) work order, the quantity on hand is decreased and the material allocation record of the work order is deleted.

This brings us to the second part of state-dependent transaction processing systems, viz. order processing. The reader may refer to the data-structure diagram of Fig. 4 as an aid for the subsequent discussion. This figure is an extension of Fig. 3. According to MRP theory work orders are either planned orders, from planned orders, or scheduled receipts. (Some systems distinguish several other types but these three are sufficient for our discussion.)

The three types of work orders are shown together at the right-hand side of Fig. 4. It should be noted that each type of work order refers to precisely one item. For planned orders, this is simply a consequence from the fact that these orders are automatically generated by the well-known MRP I algorithm. For scheduled receipts its is also required that they produce a single item type, in order to be able to generate unambiguous exception messages (reschedule in, reschedule out) for each scheduled receipt. The same holds for firm planned orders.

For ease of representation, Fig. 4 treats planned order, firm planned orders and scheduled receipts as being one entity type (work order). Generally speaking, this is incorrect. Planned orders do not have a

![Fig. 4. State-dependent data-structure diagram.](image-url)
specific identity or a unique identifier. They may be changed by MRP at any time. In regenerative systems, they are computed in one run, but not used in the next one. In net-change systems, the planned orders are maintained in the database, but there is only one manual operation possible: to make them firm planned.

Firm planned orders are quite different. They are identified by a key which is known to the user. They can be changed in many respects: lot size, lead time, exploded gross requirements, due date, and so on. Scheduled receipts, finally, are different again from firm planned orders. Exploded material requirements for scheduled receipts are initially posted as allocations. These allocations gradually disappear when component materials are issued. Value added for scheduled receipts is maintained when material issues and operation completions are reported.

All three types of work orders have a routing of actual operations, as shown in Fig. 4. For planned orders, this routing is a copy of the normative routing of the item concerned. For firm planned orders, the routing may be altered. After order releases, the routing cannot be changed any more in most cases. However, some systems allow that operations which differ from the actual routing are reported ready for a work order.

In the above discussion, work orders are implicitly considered to be internal work orders. However, similar comments can be made for purchasing orders and subcontracting orders.

Work orders are a specification of time-phased supply. For each item, the purpose of MRP is to balance time-phased supply with time-phased requirements. Therefore, we shall shortly discuss the way in which time-phased requirements are specified. Gross requirements are either dependent of independent demand. Dependent demand is generated by explosion from work orders, as we have seen. The reverse information, specifying for a particular allocation or gross requirement the source of demand, is called pegging.

Independent demand, on the other hand, may take the form of customer orders or forecasts. For the sake of simplicity, the data structure for representing independent demand is not included in Fig. 4.

Before concluding this subsection on state-dependent transaction processing systems, a few words on Just-In-Time production seems worthwhile. It should be recalled that the very purpose of the JIT philosophy is to invest in equipment and procedures which lead to small lot sizes and flow production in order to avoid expensive information and control systems. If MRP packages are used in a JIT-like production system, the data entry and consistency checks required by standard MRP becomes a real burden. Ideally, in JIT production systems, material movement within the factory is not recorded at all. In practice, the data structure of Fig. 4 remains often valid, but additional investments in user-friendly material movement monitoring procedures are required.

3. Information systems for engineer-to-order production

3.1. General remarks

Engineer-to-order production is characterized by a number of stages e.g.: tendering, product engineering, process planning (or manufacturing engineering), purchasing, component manufacturing, assembly. In its most extreme form, engineer-to-order production consists of a multi-project situation where the exact nature of the project becomes apparent only during the project itself. More specifically, each stage is only specified precisely after the proceeding stages are finished.

Thus, the engineer-to-order production situation differs from the make-to-stock situation. In the make-to-stock situation, the products and routings are given, but the precise timing and quantity of demand are uncertain. In the engineer-to-order situation, the timing and quantity of demand are given, but the precise nature of products and routings are uncertain.

Finally it should be noticed that in many order-driven production systems, the supply of materials is not a critical issue. For example in the building industry, many materials can be delivered on short notice, whereas major equipment and human resources should be planned long in advance. Consequently, it is unnatural to derive the activities from bill-of-material structures and it is quite natural to act reversely.
3.2. State-independent transaction processing systems

State-independent data in engineer-to-order information systems are at first sight comparable to what we have seen in MRP packages. Again, operations, routing etc. come to the stage. However, there are two main differences. First of all, in MRP systems these data sets were precise, complete and consistent. In engineer-to-order (and make-to-order) environments, the formal requirements for the data in the database are less strict: these data serve as an aid in engineering and configuring, and certain attributes (fields) are deliberately left blank ("white spots"). These white spots can be used to stress the fact these attributes should receive their proper value only when the customer order has been received. Therefore, the state-independent data of engineer-to-order production consists of partially complete products, projects, routings and resources. These data are called reference products and reference projects, in order to stress their incomplete nature. The projects are considered to consist of main tasks. Each task may be decomposed into a few activities, which may remain unknown up to the beginning of a task (see Fig. 5).

We shall now discuss the state-independent data structure for an engineer-to-order company in more detail. Our discussion is visualized in the data-structure diagram of Fig. 6, which is comparable to Fig. 3. First of all, Fig. 6 shows a new entity type REFERENCE ACTIVITY, connected with the entity type REFERENCE TASK. A reference with its reference activities is a kind of provisional task structure, which may be used in several projects to characterize actual tasks which have not yet been specified in detail. Reference tasks and the associated reference activities are comparable to reference routings which are found in some MRP packages. However, in engineer-to-order production a reference task is not necessarily completely specified. Furthermore, an actual task within a project may be related to several reference tasks. Selection of a specific task structure may be deferred until the moment at which an actual task for a specific customer order is started. A reference task consists of a list of reference activities. It is not required that each reference activity is completely specified, and it often occurs that activities within a task derive their attribute values from a distinct set of reference activities. The normative operation refers to a specific capacity unit.

To conclude our discussion on routings and activities, it can be summarized as follows. The information system outlined here should be considered primarily as a tool for support of a manufacturing engineer who is involved in work preparation. The support is given in a way which provides both...
flexibility and standardization. There is a sharp distinction with MRP packages, which require fully specified bills-of-material and routings, because they have to generate automatically planned orders and capacity load profiles in MRP I and CRP programs.

3.3. State-dependent transaction processing systems

In engineer-to-order and make-to-order information systems, customer orders play an important role. The problems of record keeping of past customer orders (the so-called configuration management) is interesting and important, but it would distract the attention from the main points to be made here. Therefore, we will not consider the issue here, but mainly focus on customer orders for new projects (systems).

The definition of a (potential) customer order in an engineer-to-order environment highlights some of the differences with standard products manufacturing. In MRP packages, a customer order is a list of known items to be delivered in specific quantities at a specific date. In an engineer-to-order company, a (potential) customer order is initially at best a copy of a reference network. The real work orders which will eventually accompany the material transformations on the shop floor will gradually become known. The material requirements are also at best partially known in the beginning: they become more pronounced during the product-engineering phase. The work orders (i.e. the detailed activities of a manufacturing task) will gradually become known during the manufacturing engineering stage, when the work preparation is performed. These work orders, again, are not necessarily related to an item in the item file (not even to a generic item).

If a (potential) customer order is entered in the system, it is usually related to a reference network. When the customer order is firmly accepted, a copy of this reference network is created and pegged to the customer order. This is shown in the date structure diagram of Fig. 7, which is an extension of Fig. 6. Furthermore, Fig. 7 is comparable to Fig. 4. The customer order normally has a due date. If blank
attributes in the tasks of the network of the customer order have been specified, then earliest start dates, latest start dates, etc., can be computed.

The tasks usually pass through the following stages, for example:
- not yet completely specified (unsuitable for normal network planning techniques);
- completely specified;
- firm planned (i.e. it has a specific start date and due date);
- partly ready;
- ready;
- deleted.

(It should be noticed that packages for engineer-to-order production as described here require a separate progress monitoring at the level of tasks and major capacity groups. In the first tasks of a network this is sometimes natural, because the distinction between the task level and the activity level is not very formal. However, in later stages it can be a real burden, because work-order progress does not automatically lead to an updated status of rough-cut activities.)

As mentioned, the set of activities in a task of a particular customer order gradually emerges while the previous tasks are being executed. Activities are often created by copying reference task structures. If a task in a customer order network is pegged to a reference task, it is natural to select the activity structure connected with the reference task involved. The set of activities related to a particular customer order constitutes a (detailed) network. Some packages enable some planning and scheduling facilities for activities, even when the detailed network is not yet completed, by allowing the specification of an off-set time. This is an estimation of the minimum time required between the completion date of the activity and the due date of the task to which it belongs.

Therefore, an activity passes through the following stages:
- not-yet completely specified (unsuitable for formal planning techniques);
- completely specified (latest start dates can be computed);
- firm planned;
- planned consistently with its predecessors (earliest start dates can be computed);
- open (i.e. all predecessors are finished);
- finished.

The second point to be discussed in state-dependent transaction processing is concerned with material flow recording. In MRP packages, all material flow recording is based on unique part numbers. This means that all inventory applications in MRP packages are based on part numbers. For example, cycle counting, physical storage allocation, inventory evaluation for the balance sheet, supplier performance evaluation, turnover rate estimation, computation of storage costs, to mention only a few. In information systems for engineer-to-order or make-to-order, the situation is definitely different. In these information systems, all materials other than standard items are stored and recorded by work order. This implies, amongst other things, that all the above mentioned applications have to be adapted.

4. Conclusion

This paper has argued that information systems for production management should be structured into transaction processing for state-independent data ("basic data"), transaction processing for state-dependent data, and decision support.

Databases with basic product and process information have quite a different nature for these two cases. For make-to-stock production, such as classical MRP systems, the basic information should be complete, consistent and up-to-date. This information is used by planners to generate materials and capacity plans in an automated way. For engineer-to-order production, it should be allowed that basic information is incomplete, partly inconsistent, or not up-to-date. This is due to the fact that these data are used by engineers as reference data for the creation of customer-specific solutions.

References

Johan C. Wortmann studied industrial engineering and management science at Eindhoven University of Technology. 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 on databases. Dr. Wortmann is currently employed at Eindhoven University of Technology as professor of information systems for production management and logistics.