This paper introduces the principal data processing procedures now applied within many manufacturing industries to the "requirements generation" problem.

The procedures discussed take into consideration certain related problems in production planning and inventory control.

The nature of the various problems is illustrated and flow charts for the principal procedures are included.

Requirements generation, explosions, and bills of material

by F. L. Church

requirements and explosions

Prior to the manufacture of a product which is composed of many parts, it is necessary to determine how many parts of each kind are required.[1,2] The process of determining the number of parts required to build a planned number of a given product is called a *requirements generation* or *parts explosion*. For example, to manufacture one square, jeweled box with a lid, the parts shown in Table 1 are needed for assembly (or, in manufacturing terminology, for final assembly). To build 200 such boxes, obviously 200 times each quantity listed is needed. This is the essence of requirements generation.

Table 1

Part	*Quantity*
box lid	1
box base	1
box side	4
hinge	2
screw	22
jewel	20

Table 2

Part	*Quantity per hinge*
hinge leg	2
pin	1
jewel	6

In practice, however, there are several complications. If we purchase all parts for the box except the jeweled hinges, which are assembled from simpler parts, a parts list for each hinge assembly (which becomes a *subassembly*) must be generated (Table 2). To determine the number of detail hinge parts required in 200 boxes, we must multiply 200 by 2 (hinges per box) times the quantity per hinge. The first set of parts requirements, which refer to the final assembly (the box), is called the *first level* of the explosion. The set of parts for the subassemblies (the jeweled hinges) is called the *second level*. (Each box requires a total of 32 jewels, 20 for the box, on the first level, and 6 for each of the hinges, on the second level.) The *lowest level* of the explosion is reached when no parts require further explosion.

The next complication comes from the fact that we might have

some semi-finished parts in inventory, e.g., 35 hinges in stock. In this case, even if 400 hinges are required on the first level, we would *net* our requirements by subtracting the 35 in stock and exploding 365 on the second level.

Of course our example is trivial; manufactured products often have thousands of parts, and hundreds appear on each level of an explosion.

This paper will examine the nature of the records necessary to do explosions, the possible methods of explosion, and some of the ways in which the results of an explosion may be used.

bills of material

The record of the parts that comprise an assembly description is called a *bill of material* (*B/M*). In our example, we have two bills of material. The final assembly, or *top bill*, lists the lid, base, side, hinge, jewel and screw. On the second level, there is one bill of material, giving the parts required for a hinge. Clearly we might have had more bills of material on the second level if more first-level constituents had required explosion. The logic of explosions is based upon the fact that bills of material frequently list parts which are themselves bills of material.

For our purposes, we may define a bill of material as a list of parts required to form a finished product or an assembly. A bill of material must have two basic entries for every part listed: the part designation and the quantity required. In practice, other data (to be discussed later) are also needed.

Let us consider various possible forms of the bills of material for our square box. The simplest is a parts listing of every item that goes into the product (Table 3). No concept of level is involved here. The listing has the advantage of simplicity but also this limitation: it gives no indication of the way the parts are associated or assembled to make the completed box.

When the structure of a manufactured product requires that subassemblies, such as the hinge, be noted, then several bills of material are used to describe the product. The structure of the product is then described by a nest or hierarchy of bills of material, each of which represents an assembly or subassembly. In our example, only one bill of material was part of a higher level bill. Note that the explosion is continued level by level until the product is reduced to its simplest components.

Table 3

Part	*Quantity*
lid	1
base	1
side	4
hinge leg	4
hinge pin	2
screw	22
jewel	32

Logically, bills of material form analogs of actual parts and assemblies, and are used to develop, level by level, the material structure of a product from simple relationships between bills of material. These concepts may be clarified by some concrete examples of the physical form of bills of material.

One of the common ways to keep bills of material is to store them on cards in "tub files" or on punched cards, with one card for each part. Each card would typically include the fields shown in Table 4. Field 5 is of particular interest in material explosions, since it identifies the part as (1) a detail part, (2) a purchased assembly, or (3) a manufactured assembly. For the hinge mentioned previously, the character code would indicate it to be a

Table 4

Field number	*Description*
1	part number
2	B/M number
3	quantity per bill (parts)
4	last engineering change number
5	character code
6	source

manufactured assembly and we would know another bill of material would be needed at the next level.

Possible schemes for storing bills of material for computers range from magnetic analogs of cards to matrices, but the two major categories are the record and the matrix. Record schemes are commonly one of three types:

- *Unit records,* one card or computer record for each component, as in the six-field format shown in Table 4.
- *Fixed-length computer records,* with the kind of format shown in Figure 1. Each bill of material is composed of as many records as are necessary to store all the part numbers. When multiple records are required, the address of the next record is placed in the field labeled *chain.*
- *Variable-length computer records,* similar to fixed-length records, except that chaining is not necessary because the length is determined by the number of parts on the bill.

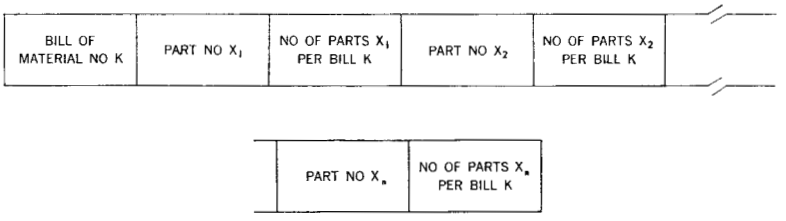
Table 5

B/M	*Parts* x	y	z
a	1	5	5
b	3	2	0
c	5	0	0
d	2	5	8

The second major kind of storage scheme, the matrix, is based upon interpreting the columns of a rectangular array of numbers as parts, and the rows as bills of material, or vice versa. Each element thus represents the number of that column's part required in that row's bill of material. For example, the four bills of material described by Table 5 (bill of material a consists of parts x, y, and z in the quantities 1, 5, and 5, respectively, etc.) could be represented by the matrix,

$$\begin{bmatrix} 1 & 5 & 5 \\ 3 & 2 & 0 \\ 5 & 0 & 0 \\ 2 & 5 & 8 \end{bmatrix}.$$

Figure 1 Bill of material—fixed-length computer record format



The matrix method is elegant and has the advantage of making the bill of material file serve as a "where-used file" (see below). A disadvantage results from the fact that if a bill of material does not use a part, a zero must be recorded. A matrix-type bill is thus composed mostly of zeros since most parts occur on, at most, a few bills of material. More efficient storage may sometimes be effected by using a separate matrix for each level. Another technique is to store only the non-zero matrix elements, each with an indicator to identify its row and column.

where-used files

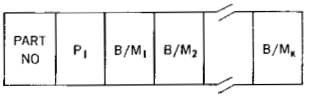
While the usual problem in explosions is to determine the parts required in a bill of material, there is also the problem of determining which bills of material include a particular part. A *where-used* record for a part is one which lists all bills of material which include that part directly. In our example, a where-used file containing the where-used records for the hinge leg and the jewel is shown in Table 6.[3] Observe that the records include only the bills that actually require the part in question directly as a part. Those bills which require the part only indirectly as a component of a subassembly are not included since the part loses its identity by becoming part of the assembly. Hence, the hinge leg does not appear on the bill of material for the box (since the hinge leg loses its identity by becoming part of the hinge).

Table 6

Part	*Where-used*
hinge leg	hinge
jewel	hinge, box

An example of a variable-word-length where-used record form is shown in Figure 2. This record indicates the K different bills of material on which part number P appears. In composing the record forms of where-used files and bills of material, observe that the roles of part and bill numbers have been interchanged.

Figure 2 Where-used—variable-word-length computer record format



When the bills of material are on punched cards or in some similar form, they can be sorted on the part number field to form a where-used file.

In a matrix-type bill of material, where-used information may be obtained simply by reading rows and interpreting all non-zero entries as where-used entries. The fact that a matrix of bills of material is also a where-used record is one of its principal advantages.

Where-used information is necessary for altering bills of material to reflect alterations in parts. For example, if we planned to substitute a bolt for the screws in our example, we would review the where-used file to determine which bills of material include screws. We would then change the bills of material accordingly.

Computer programs for requirements generation often use the where-used records to determine whether or not a bill of material is active, or to determine whether all bills using a given part have been exploded before netting.

It is difficult to choose the best method of storing product descriptions. The files must be available for the requirements generation as well as the explosions to obtain parts listings with appropriate information for cost accounting and engineering. The nature of the information required for each of these functions is

Table 7 Comparison of storage file volumes

	*Punched cards**	*Computer records*		
		Fixed-length	*Variable-length*	*Matrix*
Characters per record	80	100	—	—
W/U records	0	10,000	10,000	0
W/U characters	0	1,000,000	171,000	0
B/M records	21,030	9,600	1,407	3,337,060**
B/M characters	1,682,400	960,000	263,560	13,348,240
Total records	21,030	19,600	11,407	3,337,060
Total characters	1,682,400	1,960,000	434,560	13,348,240

* Can carry extra information not included in the other record formats.
** Matrix elements.

different. However, it is important to note that frequent changes in the parts structure of a manufactured product make one master file of bills of material essential, so that information can be kept accurate and up to date.

Table 7 is a comparative listing of the volume of characters required by various schemes to store the bill of material for a particular manufacturer. Note that the matrix elements required could be reduced by storing only non-zero elements as explained above. Obviously, since it is based on a single study, the table is merely suggestive.

part listings, requirements generations

There are several reasons for exploding a product into its detailed parts, and the nature of the explosion differs according to the purpose. Explosions may be used to obtain part listings as well as requirements generations. A *part listing* simply lists all the parts required to make a product. Part listings are of value to cost accounting, engineering, production control and many other functions. The requirements generation process determines the number of parts required to build a given number of a product, considering parts in stock or already on order. Purchasing and inventory control decisions depend on this information.

To appreciate the meaning of the information on a parts listing, one must understand the manipulation involved in obtaining such a listing. Let us then consider a computer program (see flow chart, Figure 3) for exploding bills of material for a parts listing. Assume the bills of material are stored on disks in the fixed-length record format previously described. To begin a single-product explosion (a part listing for one unit of a product), we select the top bill from the *B/M* file and first remove and store the records of all parts which are either purchased assemblies or parts which require no further explosion (the latter are called *detail parts*). Those parts which do require further explosion are hereafter referred to as *manufactured assemblies*, represented by second-level bills of material.

To continue the explosion, we call out the bills of material

for the second level and multiply the usage-per-bill field for each part by the usage-per-bill field from the level above for that bill of material. In the case of the square box, the first-level usage (two hinges per box) is multiplied by the second-level usage (two hinge legs per hinge) to obtain four hinge legs per box.

Since several parts are present on the second level, it is necessary to determine if any assemblies have two requirements records, i.e., are common parts for more than one bill of material. If so, the requirements must be combined before proceeding to the next level. (The requirements record format might be as shown in Figure 4.)

To consolidate common parts, we (1) select the part numbers that require further explosion, (2) store the requirements of all other parts and (3) sort the requirements records by part name and add the requirements fields to obtain a single summary record of usage for the assembly. The summary records whose character codes indicate they are manufactured assemblies are then used to explode the third level. This procedure is repeated until none of the parts requires further explosion, i.e., no more manufactured assemblies occur. All parts-requirement records from the various levels are then merged and summarized. (In our example, we would summarize the requirements for jewels.) The print-out would be a parts listing of the product whose top bill of material was on the first level.

Other explosion techniques might be more appropriate for different machine configurations, but the process is essentially the same: call out the bills on a level, multiply requirements, combine common parts, and proceed to the next level. At the lowest levels the parts requirements are summarized and a listing is made of all part usage.

Let us consider a slightly more complex problem—requirements generation for a product which has certain manufactured assemblies in stock. If we are developing requirements for production, the inventory level for each assembly must be checked before further explosion. The essential difference between explosions and requirements generation is this process of inventory checking, or *stock analysis at each level*, and consequently the simple explosion process becomes much more complicated.

In the square-box example, the hinge was a manufactured assembly. Before exploding the hinge into its detailed parts, the hinge requirements were netted (365) by subtracting stock available (35) from gross requirements (400). The problem of *complete netting* will be discussed later, but let us now use the term *netting* to mean subtracting stock from requirements.

Consider a program for generating the requirements for a single product. Assume that the product being exploded is the manufacturer's only product (to avoid, for the moment, the problem of common parts among *different* products). Assume further that the number of parts needed each month must be determined in order to plan production for the next year. Returning

Figure 3 Flow chart of program for explosion

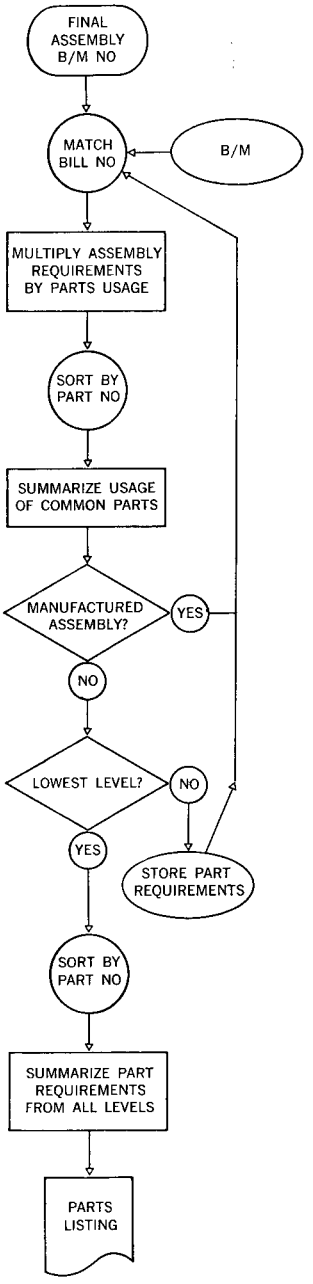


Figure 4 Requirements record for part listings

PART NO	QUANTITY REQUIRED	CHARACTER CODE

Table 8

Schedule month	*Units to be produced*
1	9
2	12
3	20
.	.
.	.
.	.
12	35

once again to the square box, we might have a schedule as shown in Table 8. The input record would appear as shown in Table 9.

On the first level of explosion, we multiply each field of the schedule record by the bill of material record field that contains the usage of part per bill, thus forming the gross requirements record given in Table 10.

Next we examine the character field for the part or parts that require further explosion (the hinge, in our example) and examine inventory to see if any are in stock. Supposing 27 hinges are available, we reduce our hinge requirements by that number to obtain the netted hinge requirements. The 27 are subtracted from the requirements for the first month, yielding 0 net requirements for Month 1, and reducing the requirements for Month 2 from 24 to 15 (see Table 11).

The input to the next level is then the netted requirements for the manufactured assemblies—only the hinge in our case. Requirements for the hinge parts are developed exactly as on the first level (Table 12). Since no manufactured assemblies appear on this second level, we know the lowest level has been reached. Accordingly, we now sort all the parts requirement records by part number. Common parts (only the jewel in our example) are summarized by adding the requirements fields for each period. The resultant summary (Table 13) gives the gross requirements from all levels.

Table 9

	Requirements by months					
Product	*Mo. 1*	*Mo. 2*	*Mo. 3*	...	*Mo. 11*	*Mo. 12*
square box	9	12	20	...	31	35

Table 10

		Quantity required					
Part	*Character*	*Mo. 1*	*Mo. 2*	*Mo. 3*	...	*Mo. 11*	*Mo. 12*
lid	part	9	12	20	...	31	35
side	part	36	48	80	...	124	140
base	part	9	12	20	...	31	35
hinge	mfd. asm.	18	24	40	...	62	70
screw	part	198	264	440	...	682	770
jewel	part	180	240	400	...	620	700

Table 11

		Quantity required					
Part	*Character*	*Mo. 1*	*Mo. 2*	*Mo. 3*	...	*Mo. 11*	*Mo. 12*
hinge	mfd. asm.	0	15	40	...	62	70

Table 12

Part	Character	Quantity required					
		Mo. 1	*Mo. 2*	*Mo. 3*	...	*Mo. 11*	*Mo. 12*
hinge leg	part	0	30	80	...	124	140
hinge pin	part	0	15	40	...	62	70
jewel	part	0	90	240	...	372	420

Table 13

Part	Character	Quantity required					
		Mo. 1	*Mo. 2*	*Mo. 3*	...	*Mo. 11*	*Mo. 12*
lid	part	9	12	20	...	31	35
side	part	36	48	80	...	124	140
base	part	9	12	20	...	31	35
screw	part	198	264	440	...	682	770
jewel	part	180	330	640	...	992	1120
hinge leg	part	0	30	80	...	124	140
hinge pin	part	0	15	40	...	62	70

Now each of these gross requirements records must be netted against the in-stock position. Assume that a product with several levels is being exploded, with several hundred parts and dozens of manufactured assemblies on each level. In such a case, the same manufactured assembly may be used in different places in the product. (For example, identical cooling fans might be used in different areas in an electronics system.) When exploding the product, such a manufactured assembly may then be required on two or more levels of the explosion.

The example given in Table 14 demonstrates the consequences of netting the requirements for an assembly before the lowest level is reached. The six records shown in the table pertain to a particular manufactured assembly. It is assumed that 100 of these assemblies are in stock. By netting the requirements shown in the first record the second record is obtained. The gross requirements of the same item on the next level at which it occurs are shown in the third record. If the requirements shown in the second and third records are combined, the fourth record is obtained. However, if gross requirements had been computed on the basis of the first and fourth records, the fifth record would result. If this record were netted against the 100 assemblies in stock, we would arrive at the sixth record. Note that the sum of the net requirements is the same in both records (the fourth and sixth) but that the requirements are in different time periods. Since one objective of inventory management is to use old stock before buying new, netting before the lowest level is reached is un-

Table 14

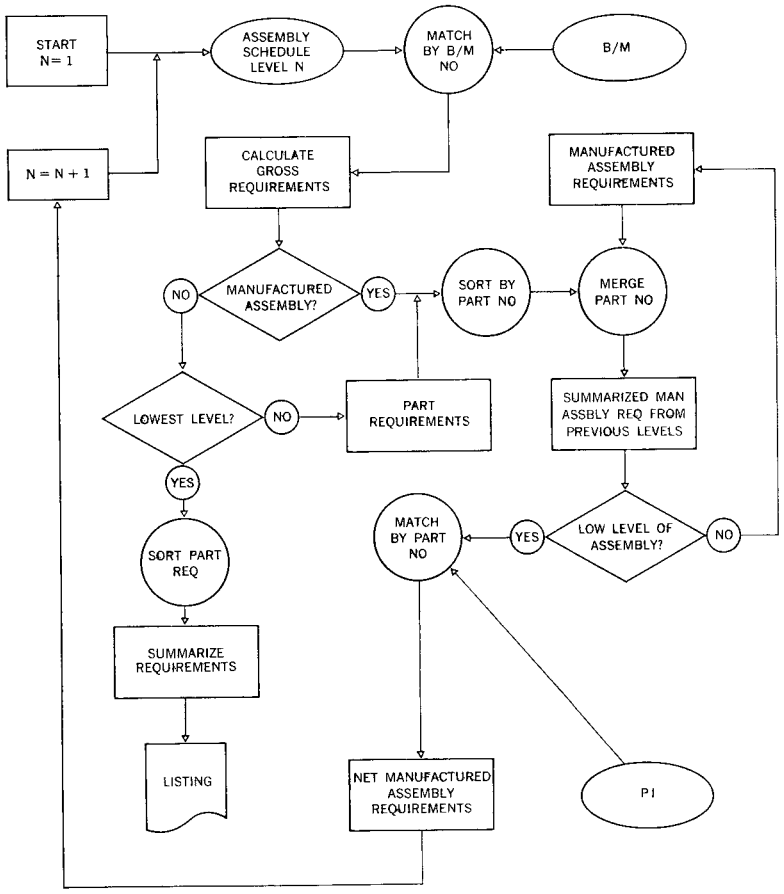
Rec.	Quantity required		
	Mo. 1	*Mo. 2*	*Mo. 3*
1	40	20	160
2	0	0	120
3	60	50	40
4	60	50	160
5	100	70	200
6	0	70	200

satisfactory. In this case, the result would be an unnecessary stock of the particular manufactured assembly in Month 1.

The procedure for exploding a commodity with many subassembly levels is identical to the way the square box was exploded, except that it is on a larger scale. A simple flow chart of a requirements generation program is shown in Figure 5.

First, the top bill of material is matched with the schedule, and gross requirements are calculated. Next, the requirements records are sorted by part number and the requirements records for manufactured assemblies are extracted. We determine which of these assemblies are *not* to be called out on lower levels and summarize their requirements from all previous levels. If the manufactured assembly is on the lowest level, the requirements are matched with the inventory records. Net requirements are then developed using the inventory records. Net requirements are found by subtracting in-stock from gross requirements, starting from period one, until stock is exhausted. The netted requirements for manufactured assemblies are then matched with their bills of material, and requirements are developed as in level one.

Figure 5 Flow chart of program for requirement generation



This procedure is continued until the lowest level is reached (see Appendix A for methods of determining the lowest level of an assembly).

At this point, all gross requirements from all levels are sorted by part number and summarized. We now have gross requirements for the product specified by the top bill of material.

To make the discussion realistic, consider generating requirements for a factory that makes several products which share some common parts. Figure 5 may be interpreted as a flow chart of a multi-product requirements generation procedure. The main difference between a single and a multi-product explosion is that the latter has several bills on the first level, each having a different schedule. The procedure is basically the same as described above.

To insure current usefulness, the explosion of requirements must be executed at least monthly and/or with every schedule revision if the results of the explosion are to be current and useful.

matrix explosions

If bills of material are stored in a matrix scheme, an explosion can be described by matrix algebra. For example, if the columns of the matrix

$$B = \begin{bmatrix} 1 & 3 & 7 & 2 \\ 5 & 2 & 0 & 5 \\ 5 & 0 & 0 & 8 \end{bmatrix} = \begin{bmatrix} B/M_1 \\ B/M_2 \\ B/M_3 \end{bmatrix}$$

are used to refer to the parts a, b, c, d, respectively, the rows may be used to denote bills of material. Suppose also the row matrix

$$E = (5 \quad 3 \quad 6)$$

denotes the requirements of the bills B/M_1, B/M_2, and B/M_3 from the previous level to be respectively 5, 3, and 6. Then the requirements matrix, R, can be found by forming the product of E and B, where "product" is defined as usual by means of the relation,

$$R_{ij} = \sum_k E_{ik} B_{kj}.$$

Thus, we would obtain

$$R = E\,B = (5 \quad 3 \quad 6) \begin{bmatrix} 1 & 3 & 7 & 2 \\ 5 & 2 & 0 & 5 \\ 5 & 0 & 0 & 8 \end{bmatrix} = (50 \quad 21 \quad 35 \quad 73),$$

and have determined the respective requirements for parts a, b, c and d to be 50, 21, 35 and 73.

While such notation is very convenient, it leaves certain problems. If the explosion goes through several levels, it is still necessary to determine on each level which parts require further explosion. The clerical work involved in keeping track of each element of the matrix might easily be more costly than keeping the bills of material in some less elegant form.

Let us explode the square box by means of matrix techniques. For convenience, we will explode for only one planning period. The bill of material matrix is shown with the rows and columns identified:

$$\begin{array}{c} \\ \text{Box} \\ \text{Hinge} \end{array} \begin{array}{c} \begin{array}{cccccccc} \text{Top} & \text{Base} & \text{Side} & \text{Screw} & \text{Hinge} & \text{Leg} & \text{Pin} & \text{Jewel} \end{array} \\ \begin{bmatrix} 1 & 1 & 4 & 22 & 2 & 0 & 0 & 20 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 & 12 \end{bmatrix} \end{array}.$$

Note that even this simple example has many zero elements.

If we explode 5 boxes and assume 5 hinges are in stock, the schedule matrix for the first level is then

$$\begin{array}{cc} \text{Box} & \text{Hinge} \\ (\ 5 & 0\ ) \end{array}$$

and the first level gross requirements are

$$\text{(Schedule) } (B/M) =$$

$$(5, 0)\begin{bmatrix} 1 & 1 & 4 & 22 & 2 & 0 & 0 & 20 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 & 12 \end{bmatrix} = (5\ \ 5\ \ 20\ \ 110\ \ 10\ \ 0\ \ 0\ \ 100).$$

In the latter row matrix denoting first level gross requirements, only one, the fifth element of the row (10), refers to a manufactured assembly. This element indicates that 10 hinges are required. Thus, the gross requirements matrix for manufactured assemblies on the first level is (0, 10).

Subtracting the in-stock matrix from the latter,

$$(0, 10) - (0, 5) = (0, 5),$$

we obtain the schedule matrix for the second level whose gross requirements are then found to be

$$(0\ \ 5)\begin{bmatrix} 1 & 1 & 4 & 22 & 2 & 0 & 0 & 20 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 & 12 \end{bmatrix} = (0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 10\ \ 5\ \ 60).$$

Since no manufactured assemblies remain in the latter matrix (the fifth element is now zero), we know we have reached the lowest level and may now add the gross requirements matrices from the two levels to obtain gross requirements, thus the gross requirements = (5 5 20 110 10 10 5 160).

To develop requirements beyond one month, the schedule matrix must contain an additional row for each period to be exploded.

Rather elaborate matrix techniques for explosions have been developed.[4,5] Our purpose here was merely to draw attention to the existence of alternative procedures employing matrices.

schedule change and selective explosions

The change of even one number in a schedule can invalidate many results in a requirements generation. Since changed schedules are frequent in manufacturing operations, requirements generations run only occasionally will be incorrect most of the time.

It is not uncommon for a manufacturer to have several hundred schedule revisions of various magnitudes in a single year. The production control department must determine how each revision affects inventory requirements. It is necessary in each case to determine whether existing part orders must be brought in earlier, or should be cancelled; whether new materials should be ordered, or whether existing orders are sufficient to cover the increased production; and whether the new schedule can even be met in the light of the lead times. To determine the effect of a schedule change, requirements might be determined by re-exploding the new schedule for all products.

However, a typical requirements generation program takes hours to run and consequently the process is costly. Furthermore, many schedule changes involve a single product and the cost of a complete requirements generation could, in fact, exceed the value of the information obtained. An alternative solution is to update the explosion whenever a change takes place by simply re-exploding those products whose schedules were altered. Such *selective explosions* can keep the requirements record file looking as though a complete requirements generation had just been completed.

Thus, it is extremely important to design a system so that selective explosions are possible. To consider this problem of being able to generate requirements for one of many products, assume that the following records are available:

- Bills of material.
- Net requirements from previous explosion requirements.
- Locator file for the bills of material.
- Perpetual inventory file (*PI*).

The records of the *PI* file, each referring to the individual parts, have a *number available* field in addition to the usual ones. This field contains the quantity of a part available (over and above those scheduled for use) during the period prior to next delivery, or during the interval prior to the next regular requirements generation (*review period*) plus lead time.

The example given in Table 15 will be used to show how the

Table 15

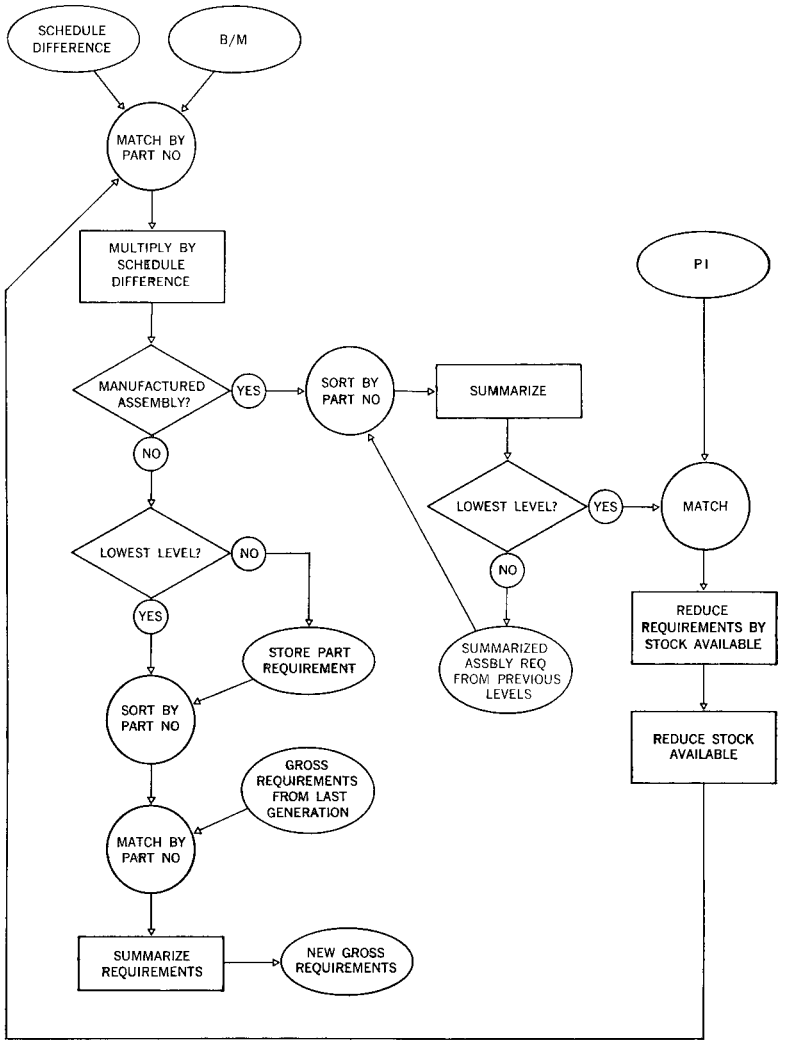
	Requirements records				*From PI file*					
		Quantity required								
Rec.	*Part no.*	*Per. 1*	*Per. 2*	*Per. 3*	*Part no.*	*Lead time*	*In stock*	*On order*	*Due date*	*Units available*
1	P_1	17	5	14	P_1	70	29	20	120	7
2	P_2	10	10	5	P_2	50	30	0	—	5
Day		80	100	120	140					

number available is determined. If on Day 80 the records in Record 1 came up during requirements generation, they would show that 7 of Part P_1 will be available in the period before the next expected delivery; 20 on the first day of Period 3, Day 120. The number 7 would be placed in the number available field. Assuming a 10-day review period, since no delivery was expected in Record 2, we would calculate the number available for P_2 to be 5 by subtracting our requirements during review period plus lead time (25 units) from the in-stock position (30 units).

If stock available is negative it should be posted. The existence of negative stock available should occasion a search for a planning error. A zero in the stock available indicates no stocking of assemblies.

The object of a selective explosion is to update the previous

Figure 6 Flow chart of selective explosion procedure



requirements generation, as shown in Figure 6, where only a change in schedule is exploded. The principal problem comes from manufactured assemblies; these must be exploded only if they were also exploded by the regular requirements generation program.

To update the requirements when the schedule is changed for a single product, the top bill of material is selected and multiplied by the schedule difference for each planning period. The manufactured assemblies that are at lowest level are extracted and matched with the *PI* file. The requirements of the manufactured assemblies are reduced by the number available, and the number available is reduced by the same amount. These netted requirements records form the input for the next level of the explosion. This procedure is repeated until the lowest level is reached. The requirements are then summarized with the requirements records from the last regular requirements generation, and the summaries are passed on to an ordering and cancelling program for analysis.

If the schedule is decreased rather than increased, excess parts should be determined and orders may need to be cancelled. The method for exploding a schedule decrease is identical with the procedure shown in Figure 6 except in the loop where the number of manufactured assemblies to be exploded is calculated. The input to the selective explosion is a set of negative values (schedule decreases).

When a manufactured assembly reaches the lowest level, it must be netted as shown by an example in Table 16. In this example (in view of a schedule decrease), the data in Record 1 was obtained from a selective explosion of a certain part at a particular level. The sum of the negative entries (−16) indicates that 16 fewer parts are now required as compared to the previous requirements generation recorded in Record 2. Since we wish to explode a maximum negative requirement in each period until the 16 (fewer parts required) are accounted for, the input to the next level is indicated in Record 3. To update the requirements records, Record 2 must be replaced by Record 4. Record 4 indicates no requirements during the first two periods. Assume these periods fall within review period plus lead time. Since entries of −4 and −2 appear in Record 1 for these periods, 6 units must be added to the number available in the *PI* file.

Table 16

Rec.	*Quantity required*					
	Per. 1	*Per. 2*	*Per. 3*	*Per. 4*	*Per. 5*	*Per. 6*
1	−4	−2	−4	−2	−1	−3
2	0	0	7	10	8	5
3	0	0	−7	−9	0	0
4	0	0	0	1	8	5

The reason for netting manufactured assemblies in the way described above may not be obvious. There are other ways in which we might have netted the requirements, but these would not have been as successful. The goal in netting requirements is to use all parts on hand as *early* as possible. By netting requirements against parts available, we have assurance that requirements will not be generated for an assembly in stock but not intended for use until after review period plus lead time.

The requirements generated in the selective explosion are exact for each part, within review period plus lead time. Beyond this period there is one case in which the gross requirements fall below the requirements that would be generated in a complete re-explosion. This low figure occurs when an assembly in stock is not entirely used up before the termination of review period plus lead time. If this occurs, then the netting by stock available in the selective explosion causes a reduction in the number of assemblies exploded for the schedule changes. Since the parts available were also netted against the gross requirements from the previous explosion as well as against the newly increased requirements, we have counted stock twice. The consequence of this error is a function of the ordering rule used; an extra purchase might be needed. But this method does protect inventory from shortages until the next explosion. At the same time, the method involves a saving in exploding only the changed part of the schedule rather than the entire product file.

The new gross requirements based on the schedule change provide other pertinent reports. As an example, a report on expected parts shortages can be obtained by matching the requirements with *PI* and subtracting the parts available from successive fields until, in effect, the parts available are exhausted. That is, requirements are netted with parts available and the number available is reduced by the same amount. If the requirements are greater than the quantity available, the number required is reduced by the quantity available and the latter quantity set equal to zero. Next we simulate deliveries by determining in what period these will be made and then reducing requirements for each delivery period until we have accounted for all deliveries. Part shortages can then be anticipated and the information can also be used as input data for the ordering program.

netting requirements

Let us reflect on the general problem of netting requirements for a detail part. The problem is to determine how many additional parts should be ordered in each period in the light of the number of units in stock and on order.

There are many ways to net inventory. The simplest is to subtract the number of parts in stock and on order from gross requirements. However, this method is unsatisfactory since it does not take into account the difference between on order and in stock; nor does it consider the expected date of delivery. A more satisfactory netting procedure is to keep on-order position by periods. A suggested part inventory format is shown in Figure 7.

Figure 7 Suggested PI record format

PART NO	IN STOCK	ON ORDER	DUE DATE	ON ORDER	DUE DATE	STOCK AVAIL.

An even more meaningful netting procedure could involve stock simulation. This would entail reducing requirements first by stock on hand and then reducing the proper requirement fields by the on-order position for that period, finally netting succeeding periods by stock that is not needed in the delivery period.

The netted requirements record is the main input to ordering programs and can also form the basis of the expedite record since positive net requirements between review time and lead time mean a part shortage. Zero net requirements for many periods ahead may indicate overordering or overstocking, and cancellations may be advisable.

To obtain the benefits of daily explosions without actually doing complete explosions, one must (1) explode completely for each planning period, e.g., each month, (2) keep gross requirements at the end of the full requirements generation, (3) keep a stock-available field for all manufactured assemblies, and (4) record the old stock position for each part in inventory. (*Old stock* is defined as: stock on explosion day minus unplanned withdrawals minus floor losses. Thus old stock is the stock-available position on explosion day less those pieces that were removed from stock for reasons not apparent during the original explosion.)

In the event of a schedule revision, one might explode only the features affected by the change and merge the changed requirements with those from the regular explosion. If the steps explained previously were followed, the benefits of *daily* requirements generation runs would be realized, i.e., all the same information would be available.

This information will serve several purposes. At the end of the requirements generation program, a set of netted requirements generation records should be formed and relayed to the program for expediting parts overage reports and ordering. A set of order recommendations, each with a date of release, is obtained from the ordering program. If the system were static, ordering would be simply on the proper day and in sequence; but changes are a way of life in production control, so that we must be able to alter the order recommendation file. It is to this end that all our efforts have really been pointed.

If a schedule is changed, all the parts affected by that change are netted against the old stock position, the results passed through the shortage, overage and ordering programs and, finally, the old ordering recommendations replaced with recommendations made in the light of the later information. A similar re-netting procedure should be followed in the event of an unplanned withdrawal from stock. By re-netting the gross requirements of the parts affected by each change, we may keep the requirements generation ac-

curate throughout the month rather than just on the day of explosion. Needless to say, accurate timely information is imperative for effective production control.

If schedules are on a monthly basis, requirements must be generated at least once a month and, on each occasion, one schedule period will be added to compensate for one dropped so that the length of the planning period will remain constant. The same pattern would be applicable regardless of the time periods involved.

Consider the problem when weekly explosions are possible, but planning is done on a monthly basis. If explosions are weekly and an attempt is made to net weekly against in-stock position, we shall encounter certain problems. Assume an explosion on the first of the month, with the results as shown in Table 17. If requirements are netted again on the 15th working day of the month, we obtain Table 18. Because stock has been used to build the products scheduled for Month 1, the picture of net requirements is false. However, the problem is easily solved by netting against the first-of-the month balance, and the motivation for defining old stock in the particular manner selected above will be apparent. The requirements from mid-month explosions are netted against old stock.

conditional planning

Usually, requirements generation is used after a decision has been made to change schedules. These programs destroy records, making it impossible to use explosions as anything but clerical reactions to decisions. Conditional planning of some kind is needed. If one can explode single products and determine how the requirements for each product affect inventory planning, explosions can *assist* in planning rather than serve only to represent the end result of planning.

As an example, consider a manufacturer of several products, each composed of several hundred parts, who is asked if he can deliver 10 more units of one product starting with Month 3. If he can explode selectively and rapidly copy his permanent records, he may be able to determine his costs for additional materials

Table 17

Gross requirements			*In stock*	*Net requirements*		
Per. 1	*Per. 2*	*Per. 3*		*Per. 1*	*Per. 2*	*Per. 3*
20	30	30	25	0	25	30

Table 18

Gross requirements for new schedule			*In stock*	*Net requirements*		
Per. 1	*Per. 2*	*Per. 3*		*Per. 1*	*Per. 2*	*Per. 3*
20	30	30	10	10	30	30

and for expediting procedures. He can determine the cost by exploding the schedule difference, developing the ordering and expediting reports, and comparing these against the same reports without the schedule change. If the change in schedule seems feasible, he simply collates the conditional plan with the rest of the schedule. If the schedule change requested seems impossible or uneconomical, the manufacturer might try an increase of 5 units per months, or some other gradual production increase. In this way explosions may be used during planning. It should be noted that the use of explosions in planning requires a method of exploding that does *not* change the permanent files.

Appendix A: Low level codings

In many of the explosions described in the text, it was necessary to know when a bill of material was on its lowest level. Let us turn our attention to this concept and discuss how the lowest level is determined and when it is important. Recall that the lowest level of a bill of material or part is that level in an explosion below which no assemblies include the bill or part. If a bill's lowest level is level N, then we explode that bill into its detailed parts on level $N + 1$. Identifying the lowest level requires looking ahead through the remaining levels of the explosion to see if the bill or part in question appears again.

The simplest way to determine when a bill of material is on the lowest level is to have a low level code as an extra field on the bill records. By way of example, let us suppose that the designated bills of material appeared in an explosion on the levels given in Table 19. In such a case, the low level codes would be assigned as in Table 20.

Now if an engineering change caused B/M_{30} to be used by B/M_{50}, the low level code for B/M_{30} becomes 5. All components of B/M_{30} must then have a low level code of 5 or greater since we do not explode B/M_{30} into its constituents until level 5. In practice, simplified explosions are made to determine the lowest level of each assembly before each requirements generation. (The time taken to execute these programs is small compared to requirements generation.)

Sometimes, random access memory is used for explosions and the requirements generation based on the total usage method. This method depends on the where-used record of each part. As each part is called out by a bill of material, the where-used record

Table 19

Level 1	B/M_{10}	B/M_{20}	B/M_{30}		
Level 2		B/M_{20}	B/M_{30}	B/M_{40}	
Level 3		B/M_{20}			
Level 4					B/M_{50}

Table 20

Bill of material	*Low level*
B/M_{10}	1
B/M_{20}	3
B/M_{30}	2
B/M_{40}	2
B/M_{50}	4

of that part is marked to note that the bill has been exploded. When all the records on a where-used record have been marked, then that part is on its lowest level. If the part is a bill of material, it is exploded on the next level. This procedure makes a low level code unnecessary.

Appendix B: Requirements generation by sorts and by random access techniques

Requirements generation based on sorting (as discussed earlier in the paper) can also be handled by random access techniques. Here, a comparison of the two methods is given.

This comparison is based on the study of a particular manufacturing facility in which the record volumes were found to be as listed in Table 21.

Table 21

Level	*B/M*	*No. of parts records*	*No. of unique parts*	*Net requirements*
1	400	6,000	2,900	1,680
2	400	6,000	1,990	1,680
3	420	6,300	3,000	1,700
4	150	2,200	800	220
5	35	500	200	25
6	2	30	30	35
Total	1,407	21,030	8,920	5,340

For each method, the number of logical and arithmetic steps for the requirements generation in this facility were determined with only those steps actually dictated by the "job" counted—steps required for program modification and randomizing being omitted. The requirements generation was assumed to be for the next twelve periods.

The sorting approach was found to require the volumes of operations indicated in Table 22. In this table, "select" refers to reading the requirements records for manufactured assemblies, which may be identified by a character code in the requirements record. "Searching" involves passing the perpetual inventory file against the summarized requirements records for netting and determining low level. The "merging" steps include sorting and merging operations on each level and the sorting of detail part requirements after the final level is reached.

Table 22

Operation	*Volume*
multiply	252,000
add	252,000
subtract	50,000
select	21,030
search	15,600
merge	281,000

A random access approach to the same problem was found to require the volumes of operations shown in Table 23. In practice, chaining and randomizing would increase the number of accesses necessary; the number given here, however, is a minimum. Also, in order to allow for order recommendations during the require-

ments generation, an additional 21,000 accesses would be needed to check the low level of each part as it appears in the explosion.

This example highlights the difference between the two approaches to explosion. The arithmetic operations are identical in number, but the operations of searching, selecting, and merging in the one case are replaced by the access operation in the other. In this example, one random access takes the place of one sequential selection and search step plus approximately nine merging steps.

Table 23

Operation	*Volume*
multiply	252,000
add	252,000
subtract	50,000
access	32,400

ACKNOWLEDGMENT

The author is indebted to R. E. Hilchey, H. G. Kolsky, R. P. Muller, and G. C. Vogel for their constructive comments after reading an earlier draft of the paper. Special appreciation is due M. F. DeFranco for invaluable advice during the study.

CITED REFERENCES AND FOOTNOTES

1. L. P. Alford and J. R. Bangs, Editors, *Production Handbook*, The Ronald Press, 1944.
2. W. G. Traub, "RAMPS: RAMAC Planning System for Production Control," IBM General Products Division Report, San Jose, California, July 15, 1961.
3. In Table 6, "hinge" and "box" are used to denote their respective bills of material. This type of "license" is used for convenience wherever the precise meaning seemed apparent.
4. A. Vasonyi, "The Use of Mathematics in Production and Inventory Control," *Management Science*, October 1954, April — July 1955.
5. B. Giffler, "Mathematical Solution of Production Planning and Scheduling Problems," IBM Technical Report 09.026, White Plains, New York, October 28, 1960.