Assessment of logistics software by means of data models applied to bills of material, routings and recipes

ELSEVIER

Computers in Industry 31 (1996) 31–43

COMPUTERS IN INDUSTRY

Assessment of logistics software by means of data models Applied to bills of material, routings and recipes

Jos C.J. de Heij *, René M.A. Caubo

University of Technology, Eindhoven, The Netherlands

Received 16 June 1995; revised 1 May 1996; accepted 1 May 1996

Abstract

The paper describes a method for evaluating software systems on the basis of their data structure. The core of the method is a Reference Data Model (RDM) with which one can assess both standard software packages and manufacturing situations. The projections give insight into the opportunities (and constraints) of the systems to be assessed. The implementation of the data structure also shows the possible (additional) functionality of a system. For illustration's sake we have applied the method to a specific functional area: bills of material, routings and recipes (BRRs).

Keywords: Data model; Software selection; Logistics software; Reference model; Standard software package; Bill of material

1. Introduction

In manufacturing companies the automation of administrative functions is often followed by the implementation of an automated production control system. If one takes into consideration the costs, full in-house development of a system is hardly a realistic option any more. Most likely a standard software package will be purchased since there are hundreds of different software packages available on the market. Hence it is important to choose the right software package and to see to a result-oriented implementation.

A software package selection used to be based on a listing of requirements pertaining to the (desired) business situation; and if possible an exhaustive list: "the more, the better". Subsequently, this checklist was sent to potential suppliers to be completed. After mutual comparison and weighing the answers a company would 'simply' choose the most suitable package.

In practice this approach proved to have major constraints. 'Shopping lists' of requirements have the natural inclination of growing all the time. For fear of forgetting something one will try and sum up as many items as possible. This leading to many detailed questions and the risk of losing track of the main issue: **what is really important**. Furthermore, processing this sort of long checklists takes up a lot of time, both of the selecting party and the software package supplier who will have to answer the questions.

The checklist approach leaves much to be desired

* Corresponding author.

PII S0166-3615(96)00034-6

if we look at its practical quality. It is difficult to formulate a question or requirement consistently, and a supplier will of course always choose the interpretation that suits him best. Finally, there comes a time – usually during implementation – when it is important to understand in which way functions have been implemented. It is true that with the aid of a checklist one can trace *which* features, functionality and options are supported, but it will be very difficult to find out *how* they are being supported.

These potential pitfalls have been recognised by most of the professional consultancy firms who regularly carry out software package selections. In order to prevent these pitfalls, a limited number of knock-out (key selection) criteria will be defined, based on the logistics controlling structure. During client-specific workshops these criteria are assessed with the assistance of an experienced package expert.

A highly promising extension to this way of working is the assessment of a system's data model, especially where it concerns the detailed assessment [3]. Although some package experts do already take into account the data model on an ad hoc basis, there is a need for a more structured approach. In this paper we describe a methodology that is based on a Reference Data Model (RDM) as developed by us.

2. Extension of package evaluation, based on a data-directed approach

In association with the Eindhoven University of Technology, Coopers and Lybrand Management Consultants is carrying out a study into the assessment of logistics software on the basis of the underlying data model. The research is based on an objective method, aimed at both the opportunities and constraints of this way of assessment.

One of the reasons to study data structures is the fact that most of the functions of an information system are of a recording nature (e.g., stock keeping or purchase, sales and production orders and their interrelationships). In addition, unlike the functionality, a data model can be described concisely and objectively. And finally, the 'functional elbow room' of the software is to a large extent determined by the data structure and an increasing amount of (program) logic is stored in the data base (stored procedures

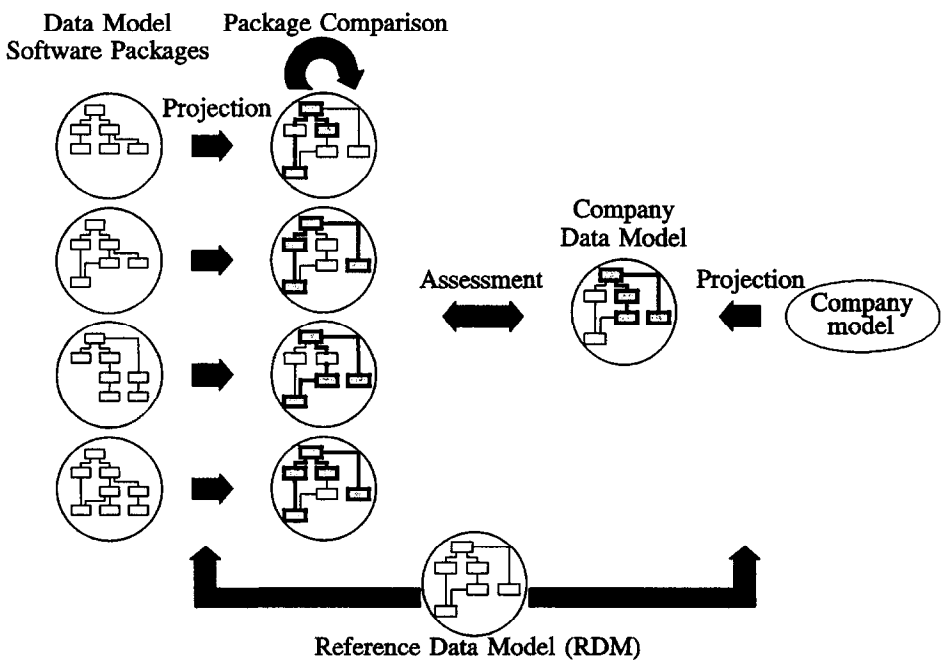


Fig. 1.

and triggers). Thus the data structure is considered as the basis (core) of an information system and any structural change in this data model will have a fundamental impact on the potential functionality. [1]

The Reference Data Model (RDM) comprises the core of our study. This model has been developed as a tool with which to compare various data models from different origins, in a similar and efficient way. Actually, the RDM serves as a kind of 'language' (esperanto) which is used to 'translate' a complex, unclear data structure into an unambiguous, comprehensible and comparable model (see also [2]). On the basis of these translations pronouncements can be made as to manufacturing systems and situations.

Fig. 1 shows the way in which a RDM is used. On the left the data models of standard software packages are 'projected' onto the RDM. This results for all data models in a 'translated' version; all using the same terminology and the same level of abstraction. On the right of Fig. 1 the RDM is also used to determine the company's data model. Subsequently, by comparing the package projections and the company's data model, the following issues can be formulated: [2]

[1] Potential functionality meaning the complete set of functions and transactions the data model of a package *could* support; whether this is current functionality or not. A package which data model, for example, distinguishes a bill of material header and a bill of material relation can easily (and probably will) support multiple bills of material per product, whereas a package which doesn't make this distinction can't. This means the right data structure is a necessary condition, but not a sufficient condition.

[2] The pronouncements which can be made do of course tell something about the RDM's quality, i.e., its distinguishing power. If, for example, an RDM does not make the distinction between capacity units and work centers it will not be possible to conclude anything about the level at which capacity can be planned. Nevertheless there is no reason to validate a RDM, as long as the pronouncements which *can* be made do suffice, any RDM is a good RDM.

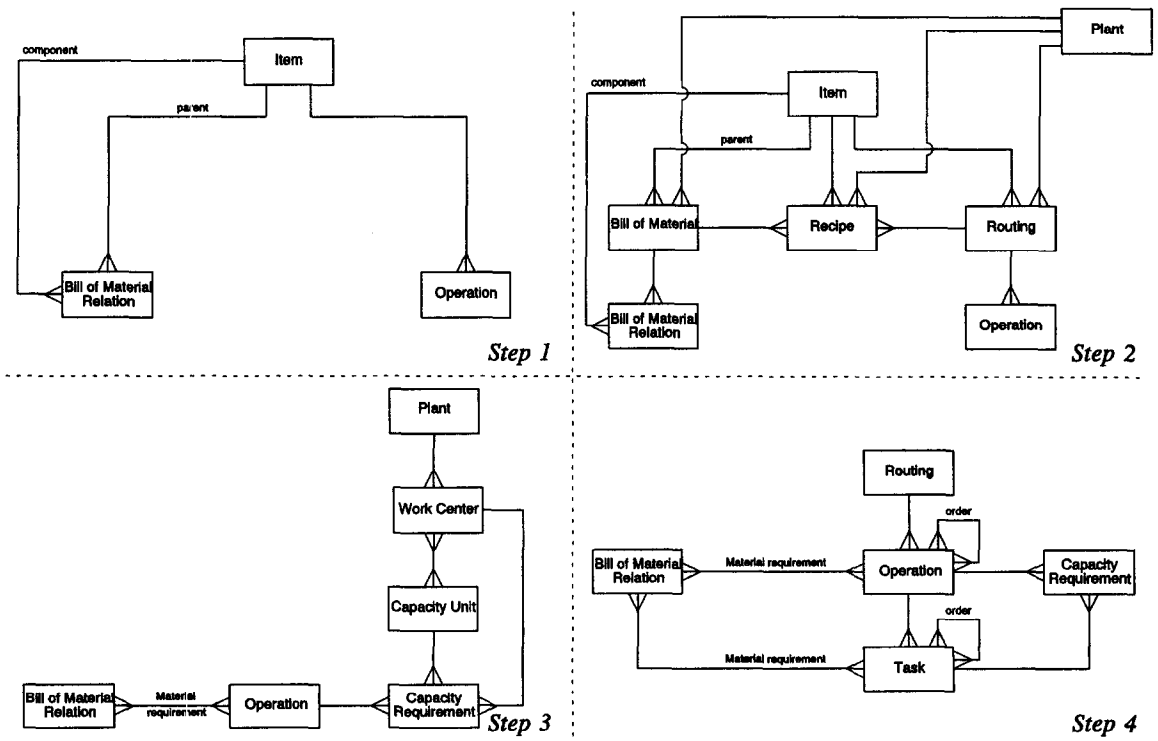


Fig. 2.

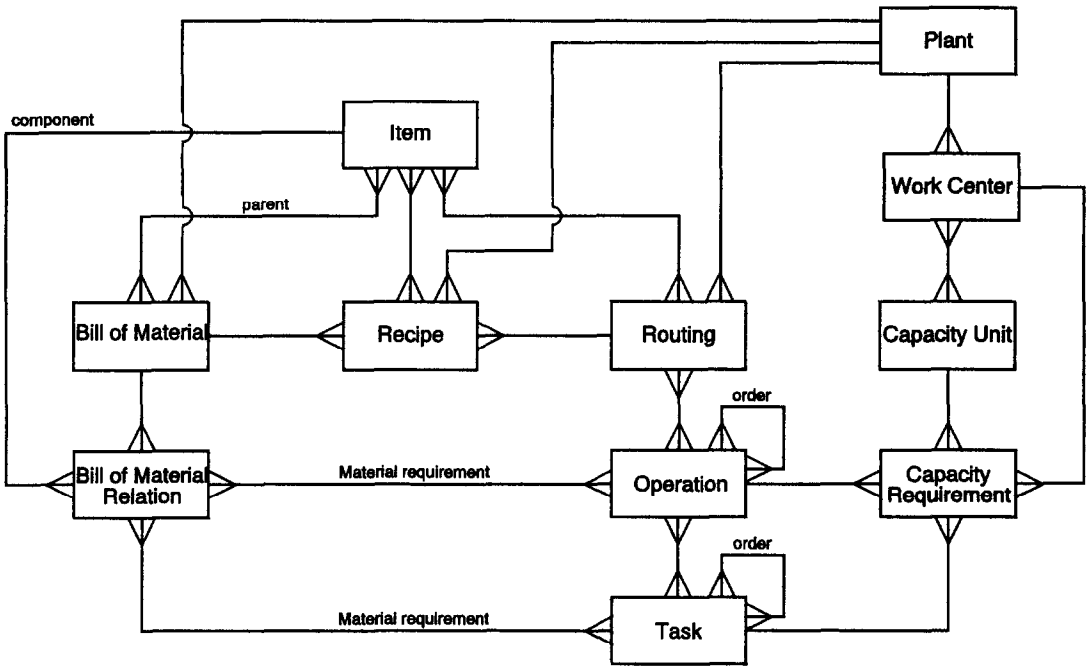

Plant
component
Item
Work Center
parent
Bill of Material
Recipe
Routing
Capacity Unit
order
Bill of Material Relation
Material requirement
Operation
Capacity Requirement
order
Material requirement
Task

Fig. 3.

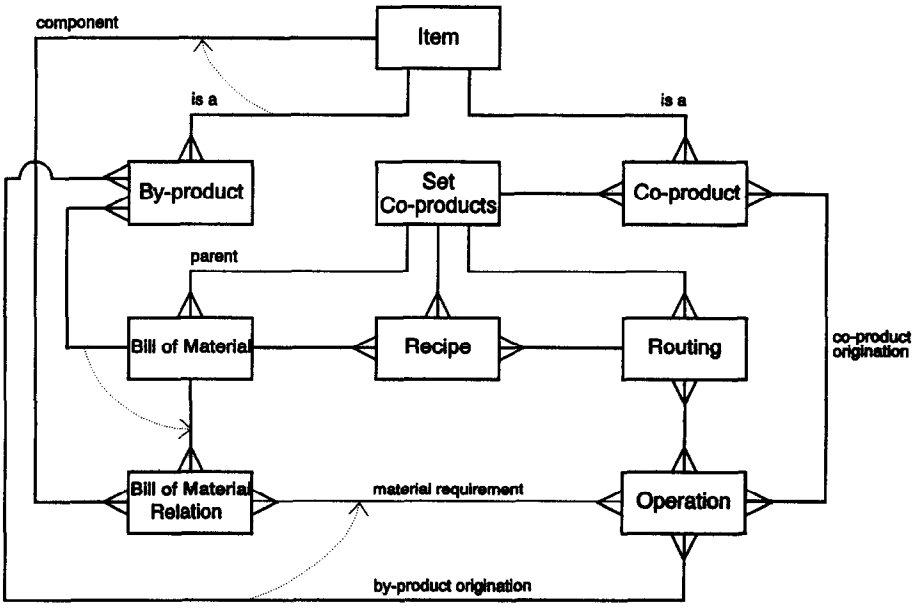

component
Item
is a
is a
By-product
Set Co-products
Co-product
parent
Bill of Material
Recipe
Routing
co-product origination
Bill of Material Relation
material requirement
Operation
by-product origination

Fig. 4.

1. how the software packages can be classified in the reference model;
2. what differences and commonalities there are between the projected packages;
3. how well the packages (or at least their data structures) might fit into the company's data model.

Of course, the projection of a package onto the RDM has to be done only once. As a projection is independent of the context there is no reason why a projection should be redone for the next selection process. Thus, after the first initial selections, the efficiency of following selection processes will increase.

Although the RDM is a sort of data model itself, it has features and opportunities that differ from the 'usual' data models. It is essential that the projections onto the RDM provide an insight in the differences between several data structures as to structure, solution and underlying concept. In the development of the RDM we have therefore made use of the data models of many packages (including the ones that where finally projected). So the RDM is actually a sort of 'lowest common multiple', extended with a vision and best practice solutions based on our business experience. Because of this 'amalgamation' of various solutions, and possibly even conflicting concepts, the RDM is not usable as the data model for a new, 'ideal' standard software package.

For our study we have divided the RDM in areas that jointly cover all logistic information flows in an industrial production company. The remainder of this paper deals with the subarea Bills of Material, Routings and Recipes (BRRs).

3. Suitability for bills of material, routings and recipes

We have developed a detailed RDM for BRRs. Part of this model is shown schematically and explained in Figs. 2–4. Because of the limited scope, this paper describes only a part of the RDM and, for example, does not go into the data structure needed for variants and alternatives.

Four well-established standard software packages have been projected on this RDM. Based on the projections, conclusions are drawn regarding the features and possible functionality of the software packages. However, the objective of this article is not to compare these packages but to exemplify the method and the opportunities of a data directed approach.

4. The reference data model

Fig. 2 shows a model for the recording of BRRs, built up in four steps and shown in its complete form in Fig. 3. The core of the model consists of the entities *'item'*, *'bill of material relation'* and *'operation'* (the terminology used in the RDM is based on the APICS definitions [1]). Using these three basic entities the structure of a product as well as the operations needed to manufacture it can be defined. These entities are therefore part of any manufacturing system.

In step 2 the entities *'bill of material'*, *'routing'* and *'recipe'* have been added. By means of the *'bill of material'* several bills of material can be defined for one product. For example, (1) to record bills of material for various business functions such as procurement, production and development, (2) to define the validity of bills of material, (3) or to define a separate bill of material for each site in a multi-plant situation [4]. The *'routing'* entity is used in a similar way to define several routings per product (a collection of sequential operations). The so-called *'recipe'* describes per site a combination of related bills of material and routings, including the way in which a product can be manufactured, and which materials are needed for production. Depending on the underlying philosophy of a software package (for example discrete or semi-process), a subset of these entities and relationships will be implemented.

Step 3 describes the material and capacity requirements at the level of each operation. With the aid of the *'operation capacity requirement'* one can define which capacities are needed for processing a certain operation. After all, one operation may claim several sorts of *'capacity units'* at the same time, e.g., a machine, a worker or group of workers, tools and other resources. The relationship *material requirement* indicates which materials or components are needed for which operation (and when). This timing feature is particularly important to lengthy manufacturing processes in which one should avoid that

materials are being reserved needlessly early. A term often used to indicate this function is 'feed-in'.

Step 4 adds another extra level to the description of the production process by means of the entity *'task'*. This type of structure can mainly be found in software packages for semi-process industries. The distinction between operations and tasks results in two levels at which one can define the capacity and material requirements. Both can be useful and are featured in packages (although not at the same time as this would lead to too much complexity and redundancy). In addition, it is advisable to classify operations or tasks of complex manufacturing processes in a sort of network structure (by means of the relationship *order*). Using these N:M relationships types one can define 'start-start' or 'end-start' relations, as well as minimum and maximum durations (like 'cure time' or 'pot life'). To combine these steps (see Fig. 3) we have extended a few relation types to the N:M relation (viz., *'bill of material'*, *'recipe'*, *'routing'*, *'operation'* and *'task'*). These extensions allow the definition of standard bills of material, standard recipes, standard routings, standard operations and standard tasks.

During the study we have detailed numerous extensions to the model. We would like to highlight one of these: the recording of co-products and by-products. These products can typically be found in (semi) process industries which often have production processes with a diverging product structure.

Fig. 4 shows the model for co- and by-products in a schematic way, including separate entity types for co-products and by-products. The definition of the distinction is: co-products are equivalent production results which are the reason for manufacturing; they thus serve as an input variable for the planning process (this is also shown in Fig. 5). Conversely, a by-product is merely an inevitable production result which may be taken into account in the planning process but which will not be leading.

As a consequence of this definition it is necessary to define co-products as extra **parent** parts in the bill of material, whereas by-products can be defined at **component** level [5]. This is reflected in Fig. 4. The additional entity type *'set co-products'* is used to distinguish the several combinations in which co-products can occur. The two relationships *co-product origination* and *by-product origination* are used in accordance with the previously discussed relationship *material requirement*. These relationships record which operations cause a co- or by-product and are therefore essential to the timing of the release of products. Looking at this data structure one can conclude that by supporting co-products, by-products could be supported as well, while the reverse is not possible.

It is interesting to observe that the entity types *'by-product'* and *'bill of material relation'* completely cover each other, including the relationships involved. This means that in terms of data structures a solution for by-products by means of a 'negative bill of material relation' is of the same value as a

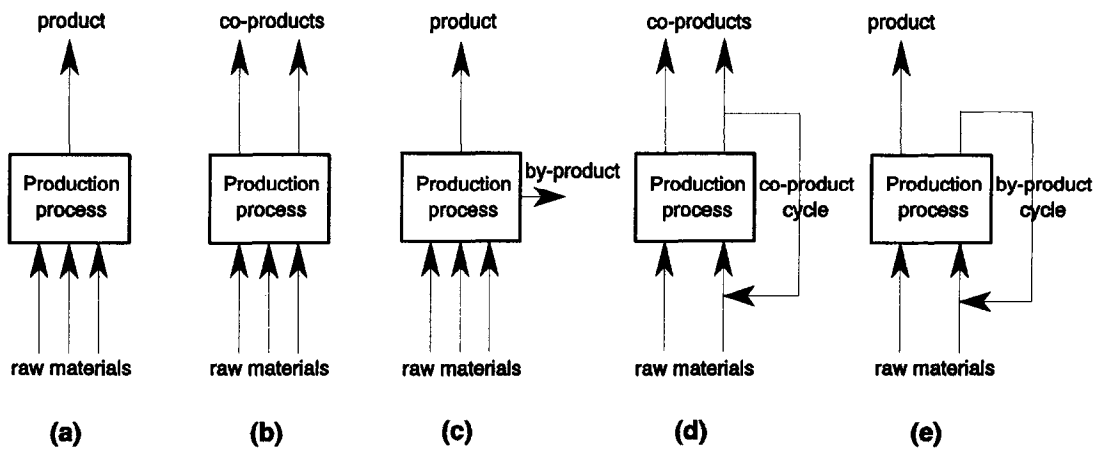


Fig. 5.

solution with a separate entity type. [3] By normalizing these negative bill of material relations one automatically creates the structure of the RDM. Naturally, this solution must also be supported by program coding in order to provide the required functionality.

5. Assessment of four software packages

The RDM as described above can be used to assess a system's functionality. In collaboration with the suppliers of some much-sold standard software packages we will discuss four examples:

1. Triton, by Baan Infosystems B.V.;
2. TXbase, by TXbase Systems B.V.;
3. Ratio, by JBA/Ratioplan Benelux B.V.;
4. Prism, by Marcam Nederland B.V.

Please note that the packages are meant for (partly) different markets, ranging from discrete to (semi) process manufacturing, and from make-to-stock to make-to-order. In view of the subject matter many of the conclusions offered will pertain to an application in (semi) process industries. After all, in most discrete manufacturing environments the requirements on bills of material and routings are highly standardised.

One should bear in mind that the actual data structure of these packages could be based on a terminology which deviates from the one used in our RDM. In those cases we translated these structures while projecting them. To guarantee the validity of our results the conclusions have been discussed with the various suppliers.

5.1. Triton

Triton is one of the most implemented standard logistic software packages in the Netherlands, supplied by Baan Infosystems. Since a number of years

[3] An old-fashioned way in which some MRP packages support by-products is the use of negative bill of material relations. Instead of *requiring* material during an operation, material is *released*. Classic examples of problems that result from this are: the timing of release which will normally be at the beginning of the operation instead of at the end, the calculation of the work in progress and the processing of reusable materials such as catalysts. These problems will have to be solved in software coding.

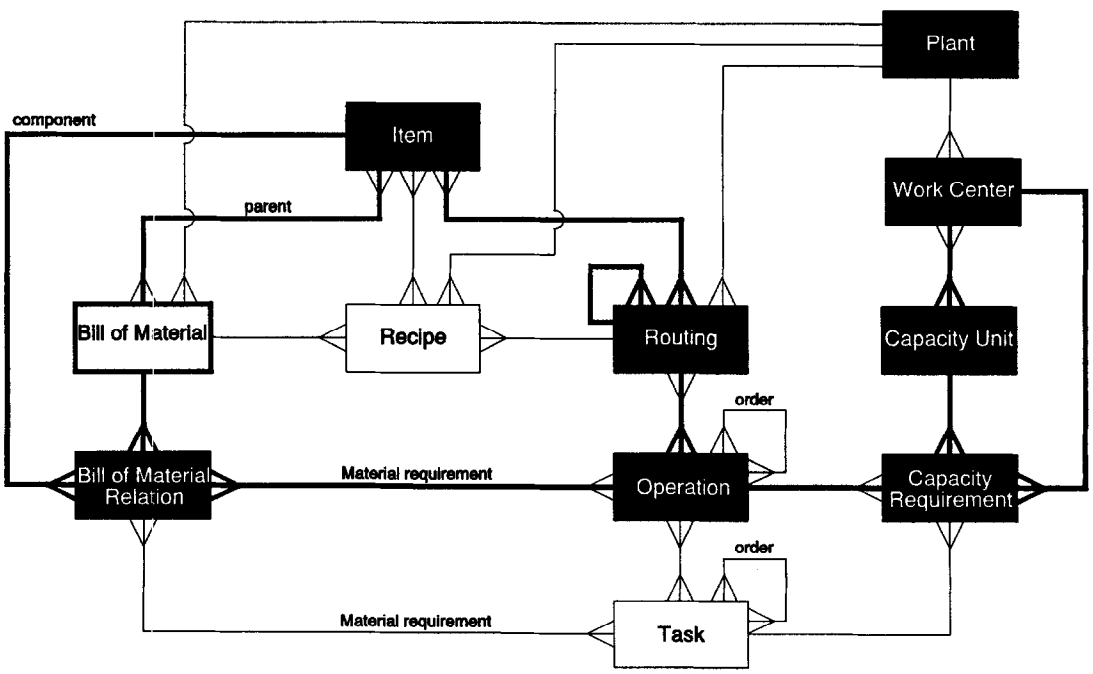


Fig. 6.

Baan has been expanding internationally. Within the scope of our study we have evaluated the Triton 3.0 release together with Baan International, the division that focuses completely on the development of the standard software package.

As may be apparent from the projection, Triton

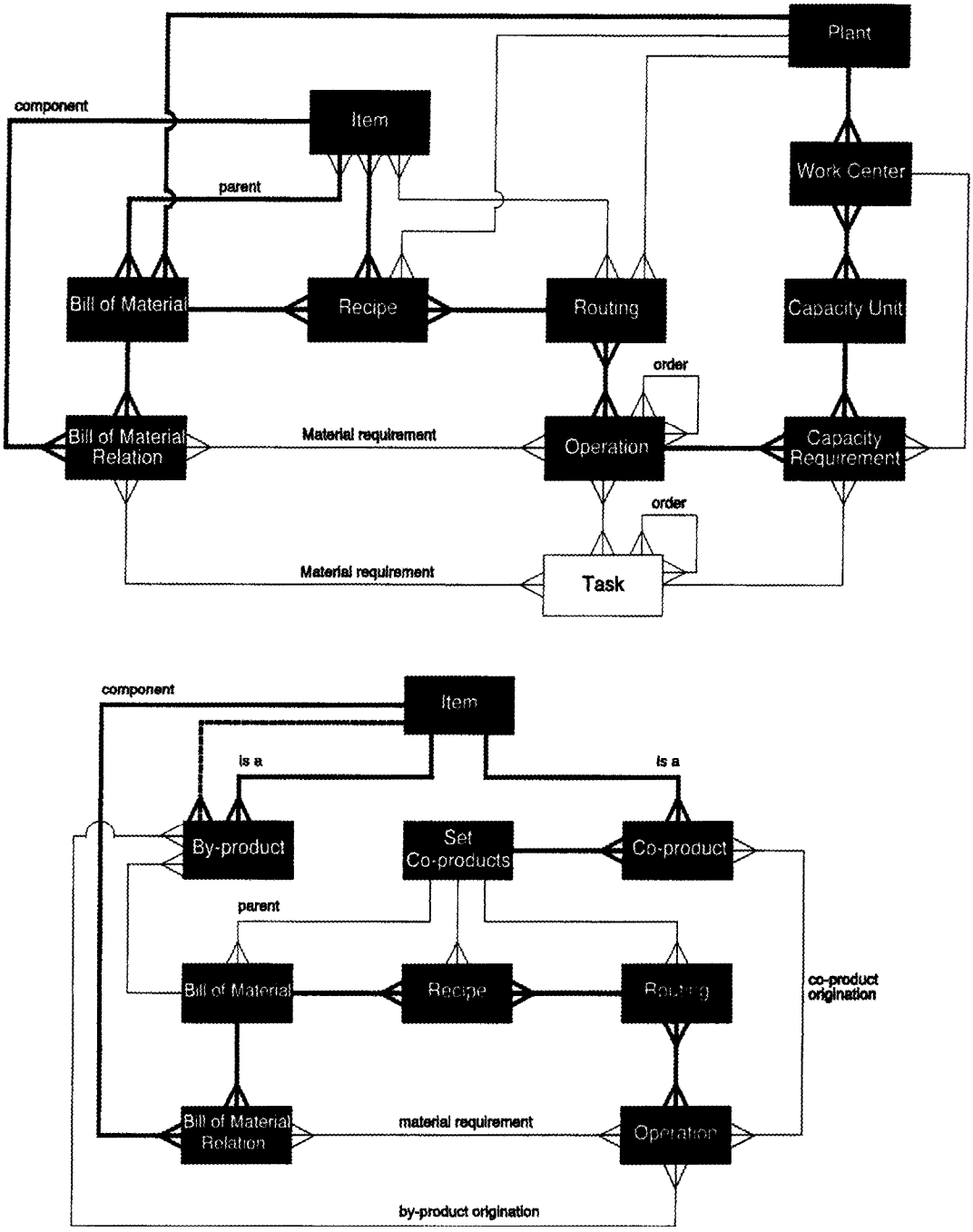


Fig. 7.

originates from the discrete manufacturing industry. Baan Infosystems has already taken steps towards the development of the package for the semi-process industry. Compared to other software packages it is interesting to describe one of the better and more widespread 'discrete' packages of this type in terms of the RDM.

Basically, Triton allows the recording of one bill of material per item. This entity has no own attributes and hence the period of validity is recorded in the *'bill of material relation'*. Outside the projected structure in Fig. 6, Triton has a separate data structure for an engineering bill of material, including drawings. The functionality of engineering changes has been implemented in this data structure. Since this aspect falls outside the scope of this paper we will not discuss it any further.

However, it is possible to record an unlimited number of routings per item, if necessary referring to a (item independent) standard routing. As regards material requirements there is a direct relation to the operation to be carried out. This allows an accurate timing – the moment at which raw materials and components are needed – in accordance with the progress of the production process. Capacity requirements are recorded per operation. Basically this concerns equipment (via the *capacity unit* relation type) and staff (via the direct relation with *'work center'*). Hence it is possible to define two types of work centers (equipment and staff). We noticed that the data model does not explicitly define a network structure of operations, although this is a typical requirement of an engineering-to-order environment for which Triton is meant.

Contrary to what one would expect from the projection (Fig. 6), the system does support multi-plant functionality. Per production location (*'plant'*) a separate data base is defined and within the application the user can easily switch from production plant. It is also possible to link entities from different data bases. The functionality of the Master Production Schedule (MPS) takes this into account. In fact, in the projection in Fig. 6 there should be a relation with *'plant'* for each entity type, but this makes the diagram virtually illegible. Co- and by-products are not relevant to the discrete production processes supported by this version of Triton, and have not been pursued by us.

5.2. TXbase

TXbase is originally a Canadian software package that is steadily gaining a foothold in the Netherlands. TXbase was developed with the help of one of the better known relational data base environments, namely Sybase. The projection (Fig. 7) shows that a routing is defined separately from an item to be produced. On the recipe level the relation lies between the bill of material and the routing. This structure is typical of the application in a semi-process environment in which a different final product is dependent on the result of, for example, different ingredients in a similar series of operations. The N:M relation type between *'routing'* and *'operation'* indicates that standard operations can be defined which can be used in several routings.

The data model shows no relation between operations and the materials required. If materials are not yet needed at the beginning of a production process but at a later stage, then this requirement should be met by the so-called 'lead time off-set' in the bill of material relation. This seemingly not very elegant solution is satisfactory in most practical situations and is found in many software packages. This also goes for the network structure of operations. Only very few manufacturing situations are so complicated as to justify this type of structural relation in the data model.

An increasing number of software packages support functionalities such as multi-site, multi-plant and multi-warehouse. In TXbase the entities *'plant'* and *'warehouse'* are each other's counterpart. According to the data model the bill of material is defined per location (*'plant'*). It is interesting to see that routings are defined across production locations. Because of the relation between the bill of material and routing via recipe one can, if need be, define a deviating routing per production location (to be monitored manually).

The capacity required per operation is defined in terms of capacity class/category (almost similar to capacity units). More or less separately TXbase has single entity types for equipment, tools and shifts (staff). We have not detailed this specialty.

TXbase is one of the few software packages with an entity *'set co-products'*, a collection of final products that are produced at the same time. How-

ever, the materials requirement is not linked to this set but to one of its items. The reason for this choice could be that this construction makes the data structure less complex and thus more practical for the many companies who have no need for it. It may look less elegant but it does function well if the software monitors the consistency sufficiently. By-products have no relation with the common bill of material. To this end TXbase has linked a separate, very simple bill of material structure (single level) to the item to be produced. This results in a simple solution, adequate in many production situations without being troublesome if the function is not applicable. Finally, we noticed that nothing is recorded about the moment at which both co- and by-products originate. In cases where timing of the materials requirement is solved by means of a *'lead time off-set'*, by-products are apparently released at the end of the production process.

5.3. Ratio

Ratio is an originally German software package that has been developed by Ratioplan. Delivery and implementation in the Netherlands is carried out by JBA/Ratioplan Benelux. Sparked off by our study the Dutch representatives have completed the documentation of the data model and included it in Ratio's standard documentation.

The data structure has some features that can be of importance in a semi-process environment (Fig. 8). Nevertheless, Ratio is not a typical semi-process software package but aims at a wider application. This becomes clear if we look at a sort of dual solution for recording recipes and bills of material (the *'or'* relations between *'recipe'/'routing'* and between *'recipe'/'bill of material'*). This means that *either* bills of material and routings can be defined per item which have no other relation than that they apply to the same item (hence no *'recipe'*) *or* that a recipe is recorded (according to a Ratio method) based on standard bills of material and standard routings that have been defined independent of an item. The *'bill of material'* itself is a so-called empty entity type of which the attributes are included in either the bill of material relation or in the recipe. In addition, various forms of bills of material can be defined, e.g., for planning, calculation, con-

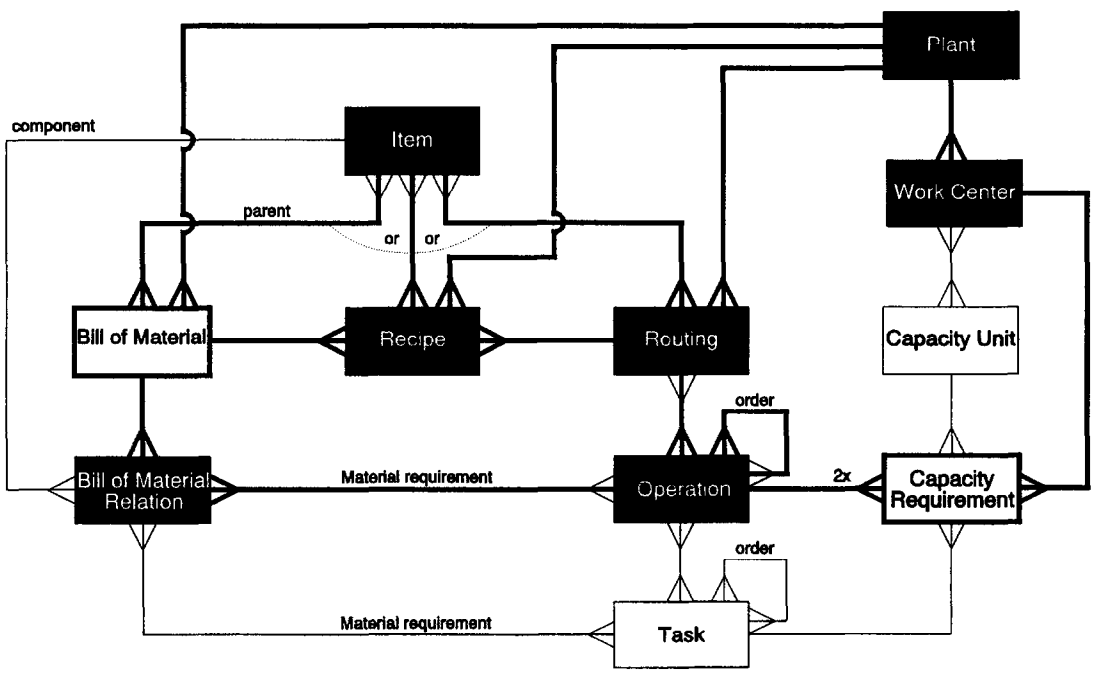


Fig. 8.

struction and manufacturing purposes.

A nice feature, explicitly captured in the data model, is the sequencing of operations with which it is possible to record convergent structures. In addition, there are attributes to indicate a so-called overlap between subsequent operations. Neither in this

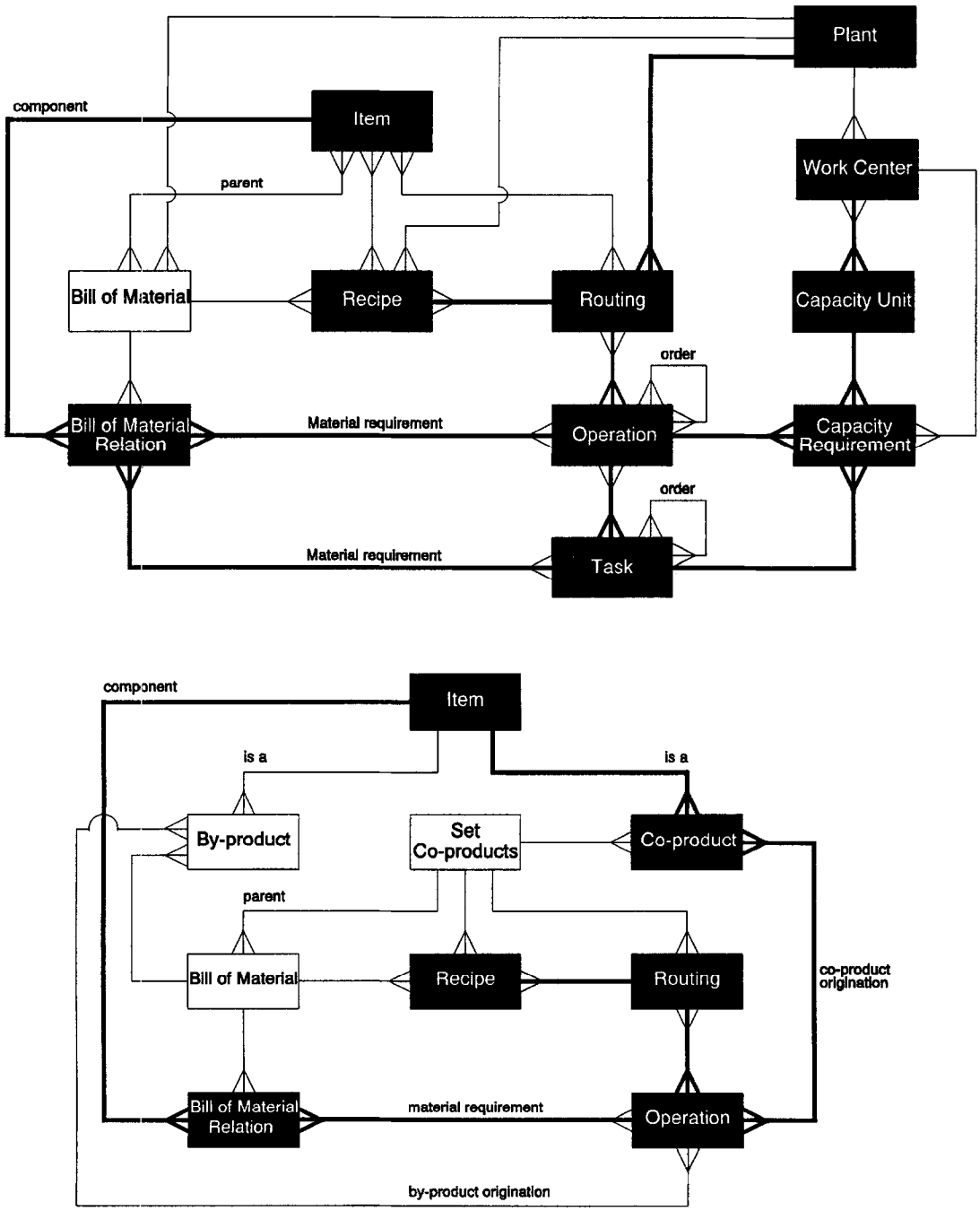


Fig. 9.

package is a complex network structure supported by the data model, but it is closer than the previous ones discussed in this paper.

In this part of the RDM the multi-plant and multi-site support is completely present. To this end Ratio employs the concepts **site–1 : N → warehouse** and **site–1 : N → work center**. There are two types of work centers: machinery (or groups of machinery) and shifts. This means that the capacity required per operation is divided in these two aspects. The tools required can be defined separately, but this speciality falls outside this part of the RDM.

By-products are supported as a separate type of bill of material relation, to be distinguished via a flag. This means that in case of normalisation the *'by-product'* entity is created. Without normalisation this implementation in the data model is similar to the negative bill of material relations.

5.4. Prism

Prism has been developed by Marcam (U.S.A.) and is supplied in the Netherlands by its subsidiary of the same name. Other than most software packages Prism [4] is specifically meant for (semi) process industries. This becomes immediately apparent from the recipe (or routing; which one is chosen for projection is not relevant in this respect) which is not directly linked to an item to be produced. Prism calls it a production model. Instead of one item as output of a recipe the items to be produced have been implemented **only** via the co-product structure. This divergent product structure is typical of some semi-process situations. It also means that if one does not need this divergence (as is the case in the discrete assembly industry), it is possible to model it in the production model. This is however certainly not more convenient than a traditional bill of material.

It is interesting to observe that three levels can be distinguished to describe the production process (Fig. 9). (For that matter, Prism has defined the terms *'operation'* and *'task'* just the other way round, but this is merely a matter of definition.) On the basis of the data model we have come to the conclusion that capacity requirements and material requirements can be defined at both middle and lower level. Prism does not distinguish between materials and capacities (and other resources) as shown in the projection by the equality of the two requirements. This means that Prism has aggregated *'item'* and *'capacity unit'* into one entity type, *'resource'*. The special power of the conceptually deviating model, based on the generalised 'Resource' entity type and the production model, is not very clear when projected on the RDM. This becomes much clearer when the data model of Prism is assessed directly, without projection onto the RDM.

The projection of the data model shows that Prism does not have by-products. We have already indicated that this is no limitation since the implementation of co-products can also be used to support by-products.

6. Conclusions

Finally, we **have not** made a list which compares all the features of the software packages dealt with above. Not only have we done so because a comparison is not the objective of this paper, but more importantly because this kind of comparison would give a false impression. All the yeas and nays listed would suggest 'the more the better'. *Instead of a resulting **list** of features the projection of the **data** structure should be part of the specification.* This should have been demonstrated in the software package discussions before.

By giving examples we have tried to show that the suitability of a software package for a special manufacturing environment amounts to more than just a list of statements like 'by-products are supported'. Such a statement would be confirmed by at least three of the four package suppliers. Nevertheless, the projected data structures show that the developers of these packages have all chosen rather different solutions. Hence, the implementation of a solution gives much more information on the suitability than can be formulated by a few simple questions and answers.

[4] The Prism concepts as described by us are subject to copyright and patent law. Violation of these rights has in the recent past resulted in legal actions by Marcam and the payment of considerable damages to Marcam by the competitive company.

Moreover, who knows what else you get when you procure a 'best score' software package? (Perhaps you won't be at all happy with all the extra functionality.) The method as shown by us makes also visible what **surplus** one buys. The importance of knowing this surplus is shown in practice where the extended possibilities of software packages can lead to enormous (initial) hidden costs. For example, all the choices to be made during the implementation and all well-meant attempts to optimise the system cost lots of time as a result of which one does not get round to the real work.

Experience with the data model approach shows that it provides the expert with a lot of insight into the elemental areas such as bills of material, routings and recipes where structures are the core. However, the approach is not one to be applied rashly or automatically. Experience and interpretation remain important aspects. In particular in the contacts with suppliers the approach has proven to be a good means to structure discussions and bring them back to the core of the issue, and to increase efficiency. To prevent any misunderstandings: it should be clear that future software selections cannot be solely based on data structures. Involving the data model in the evaluation can however bring about a considerable increase in the quality of both software selections and implementations.

7. For further reading

[6]

Appendix A

A.1. Notation technique

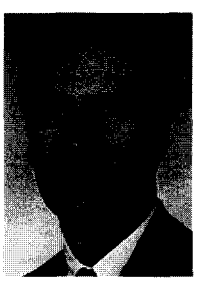
The objective being "to gain an insight" we have chosen a relatively simple notation system. The entity types are shown as rectangles and the interrelationships as lines. We have distinguished the following relationship types: 1 : 1, 1 : N and N : M. The last type has not been normalised, since the objective is not a comparison of physical data models but of conceptual ones. For the same reason we have not pronounced upon other properties of the relationships.

A.2. The more, the better?

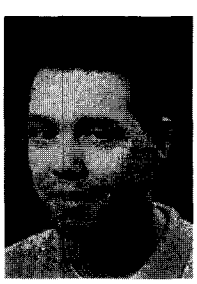
A projection of a data structure onto the RDM very quickly gives the impression that the more is covered, the better. This is certainly not the case. A software package that fully covers the RDM ("an for every situation ideal package") would definitely be a monster of a system. In order to develop an easy-to-implement and easy-to-maintain package a supplier will have to make choices regarding the concepts the package should support.

References

[1] APICS (American Production and Inventory Control Society), *APICS Dictionary*, 1992
[2] A.H. Hars, *Referenzdatenmodelle: Grundlagen effizienter datenmodellierung*, Gabler Verlag, 1994
[3] J.C.J. de Heij, "The use of data models in assessing standard logistics software", *Computers in Industry*, January 1995
[4] H. Mather, *Bills of Materials, Recipes and Formulations*, Wright Publishing Company, Inc., 1982
[5] Th.M.J. van Rijn, B.V.P. Schyns et al., *MRP in Process – The Applicability of MRP-II in the Semi-Process Industry*, Van Gorcum, 1993
[6] A.-W. Scheer, *Enterprise-Wide Data Modelling: Information Systems in Industry*, Springer-Verlag, 1989



Jos C.J. de Heij is a manager in the Manufacturing and Logistics group of Coopers and Lybrand Management Consultants. He has been working there as a professional for more than six years and has extensive experience in selecting and implementing standard (ERP) software in a wide variety of industries. Within the scope of his postgraduate research, which is the subject of this article, he is attached to the Eindhoven University of Technology.



René M.A. Caubo is now a consultant in the Manufacturing and Logistics group of Coopers and Lybrand Management Consultants. He is working there as a professional in the implementation of standard software packages, particularly SAP-R/3. As part of his graduation at the Eindhoven University of Technology he has participated in the research project as described in this paper.