A visual product constructor for engineer-to-order environments

Kai A. Olsen and Per Sætre
Molde College
Britv. 2
N-6400 Molde, Norway

Abstract

Engineer-to-order environments need a different type of manufacturing planning and control system than serial production. Since each customer order sets its own specifications it is not possible to maintain static product structures. What is needed is a tool than allows for fast creation and modification of a new product structure, where emphasis is more on information than automation. In this paper we present a methodology for such a system, which has been implemented as a prototype.

Keywords: Product structures, Product configuration, Bill of material, Engineer-to-order

1. Introduction

The traditional Bill of material structure (BOM) is effective for serial production where many units are made of each product, i.e. where the objective is to *automate*. With enhancements, a BOM structure may handle options (Vollmann, Berry and Whybark, 1992) and variants (Hegge, 1995; VanVeen, 1992; VanVeen and Wortmann, 1992a, 1992b; Hegge and Wortmann, 1991).

Variants are usually handled by a *generic* BOM, describing all possible variations of a product. The BOM for a given variant is then generated from this structure. Generic BOMs may be defined by adding conditions to traditional BOMs, by using AI-techniques (Sabin and Weigel, 1998; Yu & Skovgaard, 1998) or for example by using a programming language approach (Olsen and Sætre, 1997, 1998; Olsen, Sætre and Thorsteinson, 1996). The latter allows for greater flexibility, for automatic generation of a user interface to specify variants, and for automatic updates of product attributes (such as price, weight, etc.). The objective of these systems is to combine flexibility and automation. The limitation is that all variants have to be defined through the generic structure, i.e., all possible variants have to be known when the generic BOM is created. In practice, the generic approach is used for objects that have many common parts, and fewer individual parts.

There exist, however, several classes of *engineer-to-order* environments where the "variants" are significantly different, and will not only differ by parts, but also by design and engineering. This is certainly the case for project oriented construction activities (e.g., construction of buildings, bridges, ships, offshore installations). For some of these activities, the products may be too diversified to be described as "variants". In this paper, however, we assume certain similarities between products, that there is a commonality in design and in part types.

It is not practical for this type of engineer-to-order manufacturer to maintain a generic BOM, since each new order may describe a new, perhaps unforeseen, variant. Therefore, many of these manufacturers describe products through drawings and single-level part lists only. While the work of creating and maintaining a BOM structure for each product variant is avoided, there are several disadvantages to this approach. While drawings represent a detailed construction-oriented view, the BOM gives a production-oriented view. This is a more convenient information structure for operations such as purchase and production planning. As we will show, a *visual BOM structure* may also be used as a basis for a user-friendly interface, giving more accessible information than construction drawings.

There is another important aspect to using BOM structures for these environments. While drawings are explicit and represent the current product object, a BOM structure may represent more abstract information that may be reusable over objects. This will especially be the case if BOM structures may be created and modified by simple means.

In this paper, we present a methodology and a tool for BOM creation, the Visual Product Constructor (VPC). The objective is more on *information* than automation. Through VPC we shall try to improve all phases of product construction in an engineer-to-order environment: initial design (in order to be able to present a proposal or a bid), design, purchase, part production, assembly and service. The tool will be build around a visual BOM, the information structure for the product. All parties will use this BOM structure to store and retrieve many types of information, contracts, specification, calculations, drawings, part descriptions, etc. In fact, our idea is that the BOM shall describe the *virtual product*, the “information clone” of the physical product.

The rationale and basic methodology behind a product constructor are given in section 2, and the set of requirements in section 3. Section 4 presents the VPC tool and gives examples of its use. A discussion is given in section 5.

2. Rationale and methodology

Many engineer-to-order manufacturers have problems with their material planning and control systems. We feel that the cause of these problems is a fundamental lack of understanding of the underlying information processes in the engineer-to-order manufacturing, not only by the supplier of the MPC-system but also by the manufacturer himself.

A traditional MPC, with its focus on automation, is not what is needed in the engineer-to-order environment. This becomes apparent in BOM design. Most MPC systems consider this an infrequent process, where the idea is to formalize and present all available information in the BOM in order to allow simple and automatic processing of this structure, traversing, searching, explosion. The tools for BOM construction are not suited for a situation where a BOM has to be described for every order, perhaps for every proposal and bid. Thus, the information technology successes are few and failures many in engineer-to-order environments. Even when the implementation is considered successful, we often find that the advanced material and planning systems have been used for tasks such as finance, order handling, inventory, etc., but not for material planning and control.

Perhaps the idea of using of-the-shelf MPC-tools for these firms is wrong? The business idea for engineer-to-order manufactures is often to be “non-standard”, to ensure a niche in a competitive world market. Flexibility is needed in order to survive, to form products and production to customer requirements. Such requirements may go further than the functional specifications, as demanding customers may require a specified format for documentation, parts from specified suppliers and components marked according to their own system. The solution is then seldom a standard MPC system.

However, clearly these firms are in need of computer systems. Increasing competition requires a more efficient production. “We have to build more complicated ships in less time with reduced costs”, we were told in a local yard. In this situation, the ad hoc and mostly manual information systems break down. A system for organizing all types of information for each order is clearly needed. This system should be operative from the day the bid invitation is accepted, to be used as a basis for cost estimates, early production planning, etc. Such estimates may be based on qualified guessing, which often fails, or may be based on a product description, if this can be established at an early point in time. Of course, a complete product description will not be available before the design phase is completed. However, with a good tool it should be possible to create an initial structure based on structures from earlier products. Thus, we propose a “cut and paste” phase where an experienced designer may have the first rough BOM structure available in a couple of hour’s work. This may then be used as a basis for calculations, and – as important – as a way of organizing all information that is given for the order.

This product structure will be modified through the design and engineering phase, by modifying component attributes, adding new components, deleting unnecessary components, etc. Still, at any time the structure may be used as a basis for computing attributes and as an information source for other departments in the firm. A purchaser can therefore send orders for a finished (and accepted) substructure, even if other parts of the structure are incomplete. The purchaser will also have the choice of buying the parts and produce the (sub) structure internally, or to order the complete structure from a supplier. These decisions may be made in the moment, depending on availability, price and production load.

Since all sections in the firm work with the same structure, there will be no clear dividing line between the different phases. Parts of a product may be released for production, before other parts have cleared design. Design may, of course, also be performed after the product has left the factory, for example concerning service or modifications.

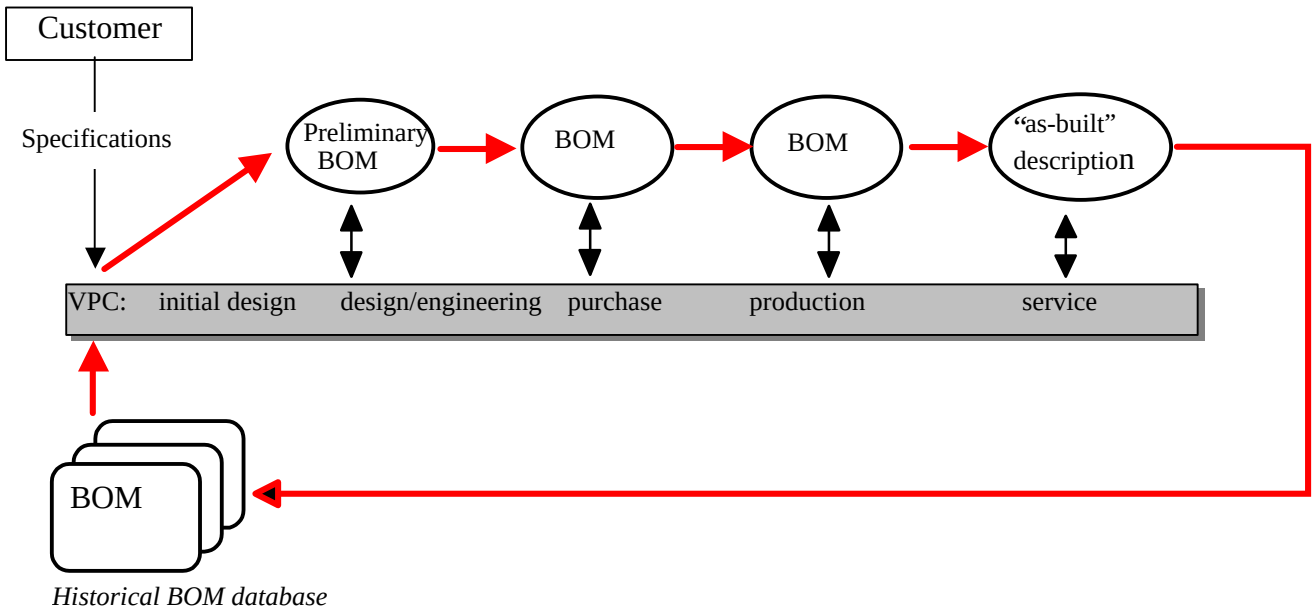

Figure 1. The virtual production line

This "virtual production line" is shown in Figure 1. An initial BOM is produced based on customer specifications and BOM structures for previous products, stored in a historical database. This structure is then modified and used through design, purchase, production and service. The system emphasizes the importance of product specification activities. From the system point of view, the virtual product is as important as the physical assembly.

The requirements for a tool supporting this virtual production line for engineer-to-order environments are given in the next section. We feel that it will be necessary to customize such a tool for each firm, due to the flexibility needed and due to the non-standard nature of engineer-to-order production. However, we do not suggest a custom design of *all* computer systems for an engineer-to-order manufacturer. While such a firm may be special with regard to products and production methods, it will have standard requirements in areas such as finance, order handling, inventory, design tools, etc. Some of these tools are stand-alone, other may easily be connected to the custom designed MPC using standard platforms (such as Windows or NT) and database interfaces (such as SQL). While simple production planning may be performed within the MPC tool, more advanced planning can be performed by transferring planning units to network planning tools (such as MS Project, PrimaVera), and to retrieve the data computed by these systems.

3. Requirements for a virtual product constructor (VPC)

Based on the previous section we may state the requirements for a VPC system for engineer-to-order environments:

- Allow easy construction of product structures.
- Allow simple revision of the information and BOM structure, from an initial draft to the final and complete producible structure. To let service retain the structure (the virtual product) after the physical product has been delivered to the customer.
- Facilitate common view of data to the different sections in the firm. Since design will be part of each order, the departments for production and purchase will have to cope with new designs and new components at all times.
- Maintain and present the complete information on an order, including the product structure (BOM).

- Maintain the status of the information (preliminary, updated, accepted, final). The component status must determine the available processes (e.g., not allowing purchase or production of a not-accepted component).
- Allow for flexibility and ad hoc decisions whether a (sub) structure should be produced in house or by a supplier, i.e., manually controlled BOM explosion.
- Support designers in selecting component types, for example to use standard types and earlier constructs as far as possible.
- Allow automatic calculations based on the product structure, e.g., cost analysis. The calculated values should therefore indicate the uncertainty in the data, e.g., which percentage of the value that is based on accepted components.

A prototype of a tool that fulfills these requirements is presented below.

3.1 Virtual product constructor prototype

A VPC-prototype has been developed using Access 97 on a Windows/NT platform. The kernel part of this system is the visual BOM, a tree structure describing the structure and the parts that goes into a product variant.

We use a mountain bike as an example, since this represents both a simple and a common product. Normally we expect that a bike is produced as models or perhaps as custom specified variants, but let us assume that our engineer-to-order bike manufacturer allows the customer to also influence the design of the product.

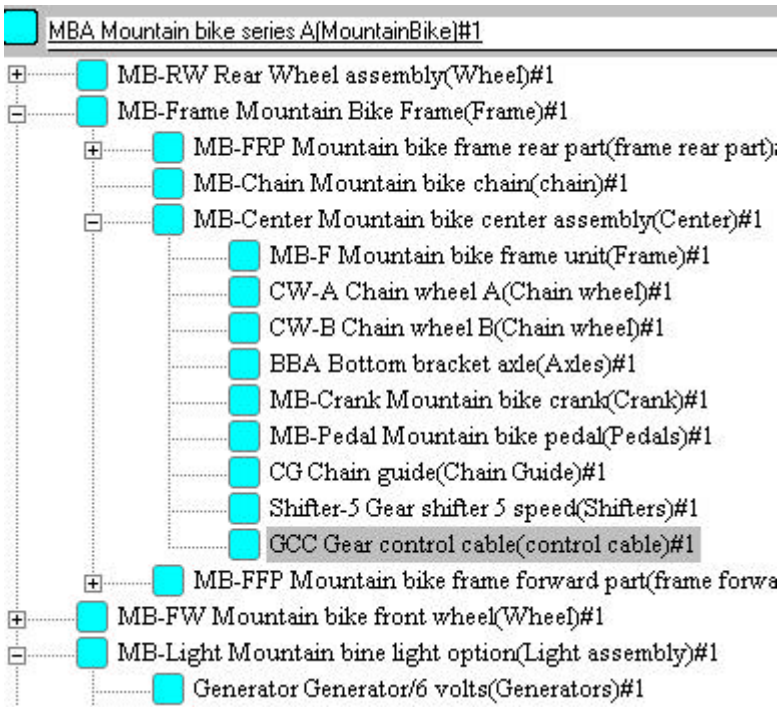

Figure 2. Virtual BOM (example).

The BOM for such a mountain bike is presented in Figure 2. Each node (square icon) represents an item, an individual component. No distinction is made between higher level and leaf items. The structure may be expanded by “clicking” on the ⊞ button, condensed by using the ⊟ button. The form and color of the node icon will present the status of a node,

for example: created nodes (blue rectangle), nodes copied from another structure (yellow rectangle), modified nodes (a rectangle within a rectangle), accepted nodes (green), purchased nodes (a P- icon).

3.2 Component types and classes

Complex products are produced from components where each component can again be a complex construct. To facilitate the reuse of both complex constructs and basic components, it is important to describe them in terms of the attributes, which are relevant in the setting in question. A given component belongs to the product tree structure. But it is also designated to a component type of a given class, which is described by its set of attributes. For a producer of mountain bikes, examples of classes may be wheel, frame and saddle. The attributes of the class wheel might be diameter, weight, material and strength. The construction support system therefore must include tools to describe both component classes in terms of attributes and to describe component types in terms of values of the class attributes.

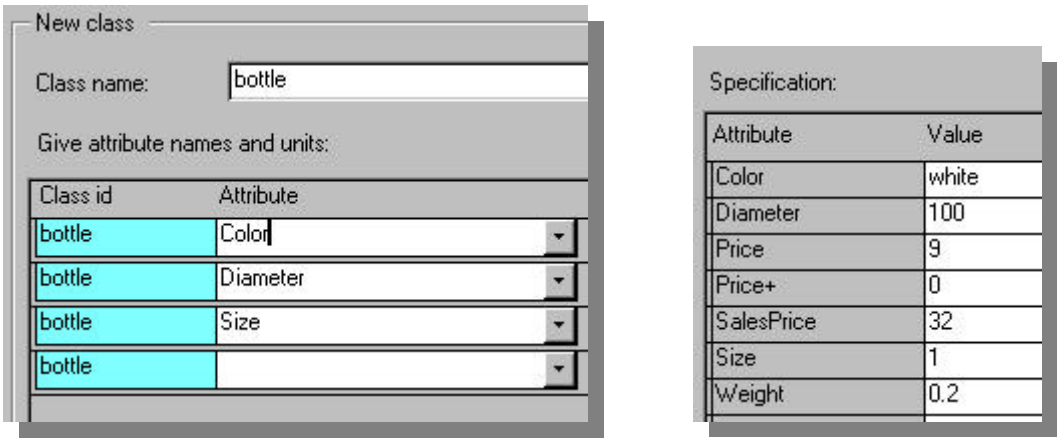

Figure 3. Class definition (left) and component type specification (right)

Component classes and types are user defined. An example is seen in Figure 3 (left), where a new class *bottle* with three attributes is defined. Class attributes are chosen from a list of predefined attributes, to ensure standardization over attribute names, units and value types. A type (object) of this class may now be defined by giving a value for each attribute (Figure 3, right). For example, we may declare the type *water/1*, defining a type for a one-liter water bottle. Note that in addition to the attributes specified by the user for the class bottle, the system will add attributes from a set of super classes. These super classes have attributes (such as price, no. in stock) that pertain to all items. In this way, the system supports the concept of multiple inheritance, as know from object-oriented terminology.

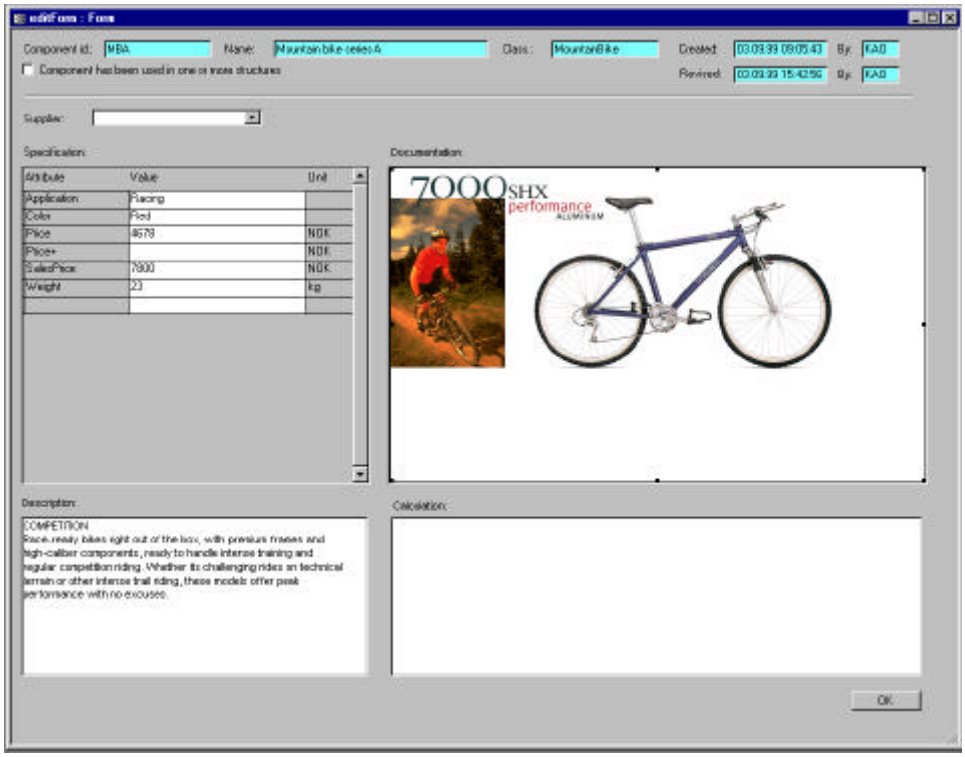

Figure 4. Component type information (example)

In addition to specifications, the system accepts information in the form of textual descriptions, pictures, drawings, spreadsheet calculations and other documents. This information is stored together with the component type specifications in a set of default attributes (*text, documentation, calculation*), as seen from the definition of a mountain bike type in Figure 4.

3.3 Creating the initial structure, editing

An initial BOM structure may be defined starting with a blank structure, and inserting single items. However, normally it will be much faster to copy items from existing structures. These may be generic descriptions (for example a generic frame part), or structures from previous products. For example, the new product may be put together with sub structures from several different product variants.

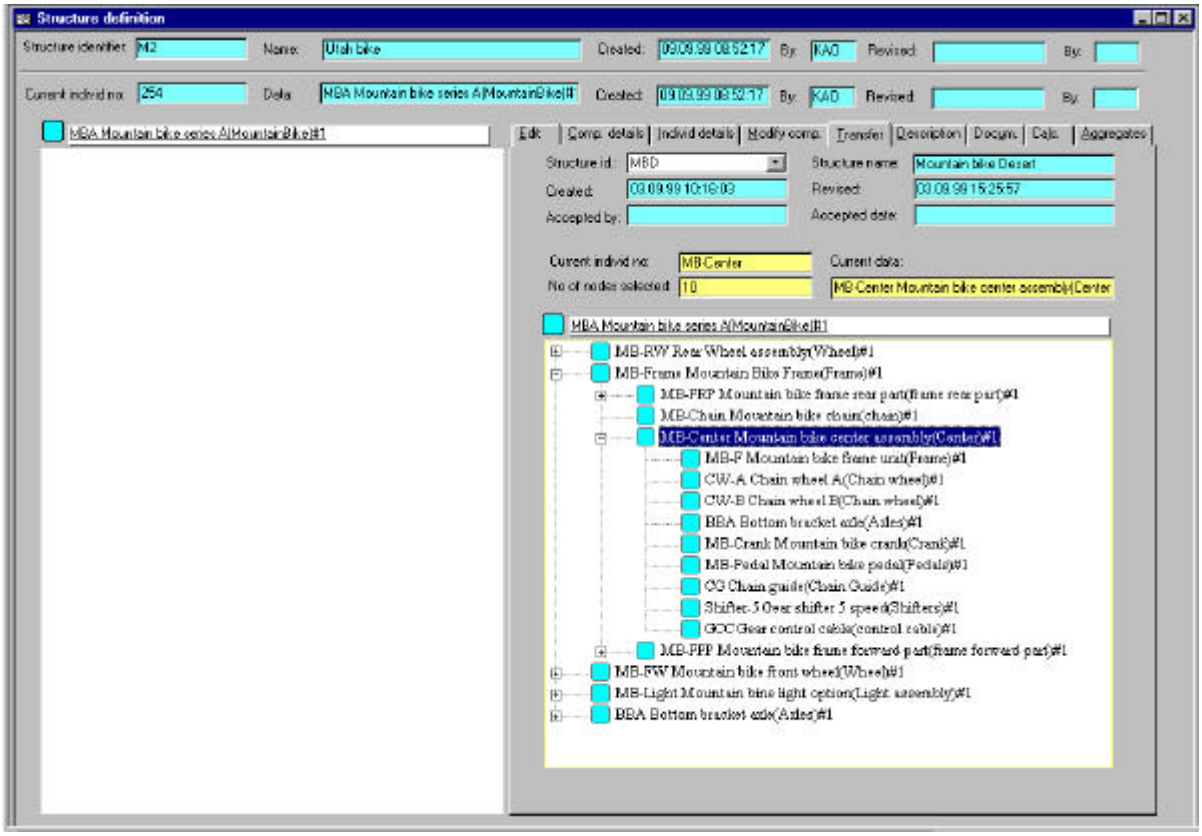

Figure 5. Transfer of items from a historical to a new structure (example)

An example is shown in Figure 5. The new structure is to be constructed in the left frame, and a historical structure is shown in the right frame. The user may click on any item (sub structure) in this frame and transfer the item (and its sub items) to the new structure. Copied nodes are marked with a special icon, and historical data (when the item was created, from where it came, date and time) are retained.

New nodes may be inserted anywhere in the structure. The insert command requires a component type identifier and the number of items. For example, to “insert” a water bottle (perhaps in a holder on the frame) we may click on the *mountain bike frame forward part*, before we give the insert command. The water bottle item will then be inserted as a child of this node. Similarly, nodes may be deleted, copied or moved within a structure.

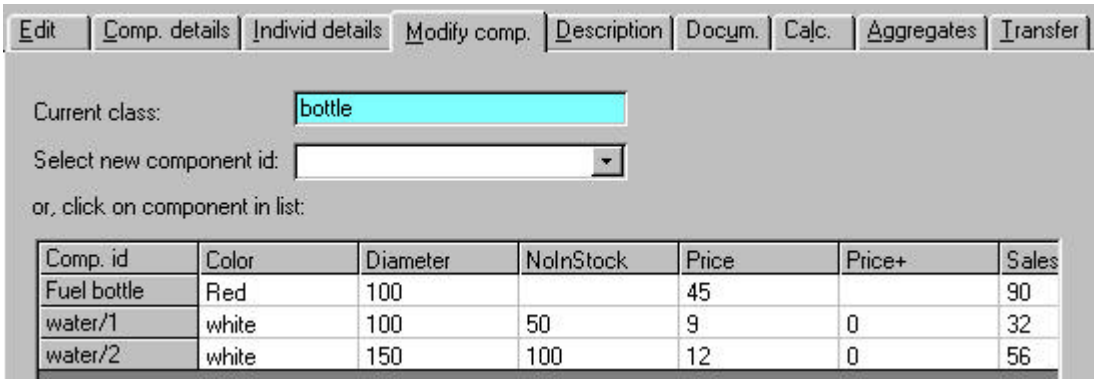

Figure 6. Change of node component (example)

The component type of a node may be changed to another component type of the same class. In these situations the system will always present all available components of the current class. This is seen from the example in Figure 6. The user has marked the water bottle node in the structure, and chosen the “modify component” tab. A list is then given of all component types of this class. An existing type may then be selected.

Alternatively, if none of the existing types are satisfactory, the specifications themselves may be changed. This implies the creation of a new type, as explained in the next section.

3.4 Node approval

As discussed, nodes may be copied from other structures and their specifications may be changed at any time. Further, the node structure is open for extensive editing. Therefore a node approval routine is necessary. A user with sufficient credentials may *approve* a node using a specific command. If the node has children, all of these must be approved before the mother node. If the component specification has been modified, a new component type will be created. The new component type will be added to the component archive, where it will be available for other structures as well.

Approved nodes will be marked with an approval-icon. If a node is modified after approval, it will return to the previous status. All approved higher level nodes will then get a new “approved?” status.

3.5 Aggregations

All nodes inherit attributes from the super class *aggregation*. In our example class aggregation has the following (user defined) attributes:

Price	- component price
Price+	- the cost of assembling a component
Weight	- component weight

These attributes will be used for aggregating over all nodes in the selected structure.

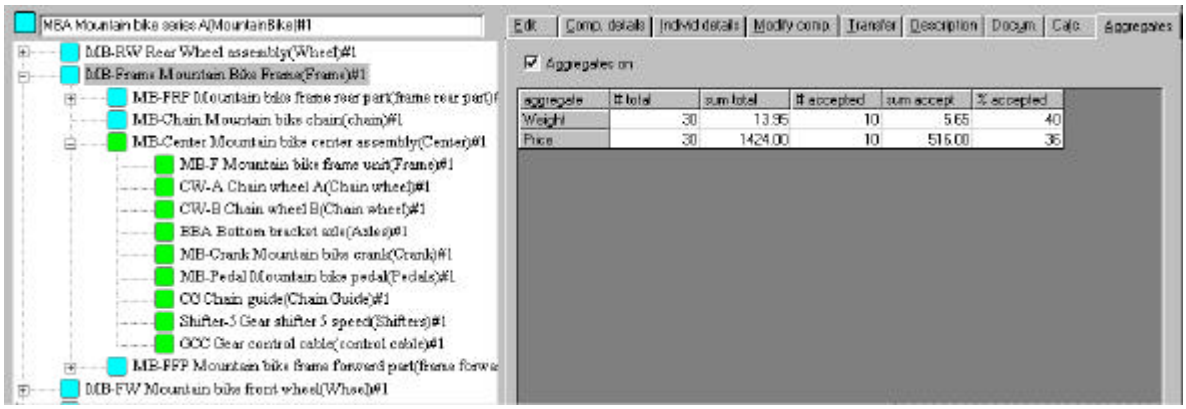

Figure 7. Aggregate values presented for different node status types..

An example is shown in Figure 7, where the aggregates price and weight for the mountain bike frame is presented. Price will be the sum of prices for all sub components, including the assembly price (price+) for all non-leaf nodes. Values will be given as totals, and for

approved nodes. As seen from the last column in Figure 7 only 36% of the price is based on approved data.

3.6 Purchase and production

When a BOM sub structure is complete and approved, it may be used for purchase or production. A production command will produce a parts list of all components in the sub structure, to be used as the basis for purchase or production orders. Specifications, down to detailed drawings, may be enclosed with each order.

The system will keep track of lead times and production dates. Where only simple planning is needed, this may be done within VPC. If preferable node data may be exported to a more advanced planning tool, for example for network planning. Using a standard platform this export may be performed in a seamless manner. Similarly, dates and other data may be imported into VPC from the planning tool.

In an engineer-to-order situation, there will be no clear distinction between design and the production phase. For example:

- To simplify design, parts of the structure may not be included in drawings, etc. Specifications of fittings, cables or hoses that connect various parts of the product, may be included more easily after the preliminary assembly.
- If the structure described in a drawing turn out to be awkward to produce, one may decide (on the shop floor level) to implement the desired functionality in a different way. In some cases, this may require a revised design, in others, the production department may do the modifications on their own accord.
- If component specified by the designer is not available, a purchaser may decide to use a different component with better or similar specifications.

In all these cases, it is necessary to add the new information to the product specifications in order to keep a one-to-one correspondence between the virtual and physical product. Thus, the component description system must be available everywhere in the organization.

3.7 Service

When the physical product leave the factory, its virtual description remains. In this way, a complete as-built description is available. The as-build description used by the service department to obtain component and structure information will be modified when components are replaced. Design may even be performed after the physical product has been finished, e.g., if the customer require a major upgrade.

4. Discussion

We have presented a tool for engineer-to-order environments where the emphasis is more on information than automation. The kernel of this tool is a visual product structure. This may be created by copying components and structures from similar products, may be edited by a direct manipulation interface throughout the whole production process and may be used as an archive structure for storing all types of product information.

In opposition to a traditional MPC system VPC may present and work on incomplete structures. That is, a product structure may be available from the very start of an order, perhaps even before the order is received (supporting a bid phase). Even if incomplete, this structure may be used for processing parts list, estimating prices and other product attributes, as a basis for requesting offers from suppliers, etc. The accuracy of the data given will increase as the structure is revised and approved. The exact structure and the final numbers will not be available before the whole structure is approved. However, then the information may be of interest only for the accountants. At this time, contracts have been signed, purchase orders given and production may have been started. Thus, the initial and intermediate structures will be most interesting for decision making. It is therefore important to have a system that allows the use of any type of information, even if it is not 100% reliable and complete.

Integrity is maintained by an approval process, where single nodes or substructures may be approved by users with the sufficient credentials. Data on an approved component or substructure may be used for purchase and production orders, even if other parts of the structure are yet incomplete.

The system will invite the designer to use already existing components or components types, by making it easy to retrieve specifications for all components of a given class. The class concept makes it possible to structure components on a higher level than type, i.e., it encompasses all component types with a given set of attributes or a given functionality.

The VPC tool presented here is not meant as a complete of-the-shelf tool. Due to the diversity inherent in engineer-to-order manufacturing, we expect that each firm will have their own requirements for the specifications and implementation of a VPC-product. For example, in some cases planning may be performed within VPC, in others the VPC must interact with a standard planning system. While these firms, due to their diversity, have problems using standard MPC tools, they will have very similar needs with regards to tasks such as finance, accounting, stock keeping, etc. Instead of incorporating these features in the VPC tool and thus developing a complete MPC tool, we develop VPC to work together with such tools.

In this way, VPC will act as “glue” between a set of programs. VPC handles the specific requirements of the engineer-to-order manufacturer, while of-the-shelf programs handle the more standardized tasks. Experience shows that it is possible to combine different programs into a system, which acts like a single system from the user point of view, using standardized interface components, e.g., OLE-links in Windows and SQL for data transfer.

Many firms have been negative to a strategy based on development of customized software. Earlier experience with costly projects acts as a warning. We have tried, successfully in some cases, to argue that modern tools make it easier both to develop and maintain software and that the connectivity mentioned above makes it unnecessary to develop everything, that different systems may be linked together in a seamless fashion.

The main argument, however, for the need of customized products is *competitiveness*. Many engineer-to-order firms that we have worked with, exist due to their capability in making ships, offshore cranes, propeller systems or articulated dump trucks based on customer specifications. While more and more of the simple processes, hull construction,

welding, sub part construction, is performed by sub contractors, it becomes clear that their competitiveness is in their ability to organize the work, a task that is especially difficult and important in engineer-to-order environments. Only the firms that manage to organize their work in an efficient manner will survive in a global market. IT tools will be a major factor in achieving this goal.

5. Conclusion

A methodology for creating product structures in an engineer-to-order environment has been presented. Through a cut & paste process a new structure may be build in a very short time. Even when incomplete, structures may be used as a basis for calculating cost and other aggregates, for presenting component lists, etc. A structure is presented visually as a node tree. This visualization also acts as a user interface for modifying the structure, i.e., operations as insert, update, delete, move and copy may be performed by direct manipulation on the nodes.

All types of product information, specifications, documents, drawings, calculations, etc. may be stored in this virtual product structure. Complete and approved parts of the structure may be used as a basis for purchase and production orders, i.e., in constructing the physical counterpart. When the physical product leaves the factory, the virtual remains. This may then be utilized by service as an as-built structure and by design when new similar products are to be constructed.

The ideas described here have been implemented in a prototype system.

References

- Hegge, H.M.H. (1995). Intelligent product family descriptions for business applications, *Ph.D. thesis*, Eindhoven University of Technology, The Netherlands.
- Hegge, H.M.H and Wortmann, J.C. (1991). Generic bill-of-material: a new product model, *International Journal of Production Economics*, 23, 117-128.
- Olsen, K.A. & Sætre, P (1998). Describing products as programs, *International Journal of Production Economics*, 56(1).
- Olsen, K.A. & Sætre, P (1997). Managing Product Variability by Virtual Products, *International Journal of Production Research*, 35(8), 2093-2107.
- Olsen, K.A., Sætre, P. and Thorstenson, A. (1996). A Procedure-Oriented Generic Bill of Materials, *Computers & industrial engineering*, 32(1).
- Sabin, D. & Weigel, R. (1998). Product Configuration Frameworks – A Survey, *IEEE Intelligent Systems*, July/August.
- Yu, B. & Skovgaard, H.J. (1998), A Configuration Tool to Increase Product Competitiveness, *IEEE Intelligent Systems*, July/August.
- VanVeen, E.A. (1992). *Modelling product structures by generic bills-of-material*, Elsevier Science Publishers, Amsterdam.
- VanVeen, E.A., and Wortmann, J.C. (1992a), Generative bill of material processing systems, *Production Planning and Control*, 3 (3), 314-316.
- VanVeen, E.A., and Wortmann, J.C. (1992b). New developments in generative bill of material processing systems, *Production Planning and Control*, 3 (3), 327-335.
- Vollman, T.E., Berry, W.L., and Whybark, D.C., 1992, *Manufacturing planning and control systems*, Business One Irwin, Homewood, Illinois, USA.