Designing workflows based on product structures

W.M.P. VAN DER AALST
Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands, telephone: -31 40 2474295, e-mail: wsinwa@win.tue.nl

Abstract

Workflow management systems (WFMS) facilitate the everyday operation of business processes by taking care of the logistic control of work. Business processes supported by a WFMS are *case-driven*, i.e., tasks are executed for specific cases. A case corresponds to a service to the environment, e.g., approving a loan, processing an insurance claim or handling a traffic violation. A case corresponds to a *product* that needs to be produced. Although the product is not a physical object, it has an internal structure, i.e. it is an informational object assembled from components. Therefore, the well-known Bill-Of-Materials (BOM) can be used to describe the product that is manufactured using a WFMS. This paper describes a technique to automatically generate a workflow process based on a BOM. It allows workflow designers to think in terms of the end-product instead of the internal process.

Keywords: workflow management; bill-of-materials; product structures; Petri nets; analysis of workflows

1 Introduction

From a logistical point of view, there are many similarities between administrative processes and production processes. Both kinds of processes, focus on the routing of work (workflow) and the allocation of work to resources. In a production system, the products are physical objects and the principal resources are machines, robots, humans, conveyor belts and trucks. In an administrative process the products are often informational (e.g. documents) and most of the resources are human. Although there are many similarities, there are also some logistical aspects in which an administrative process differs from a typical manufacturing process:

- making a copy is easy and cheap (in contrast to making a copy of a product like a car),
- there are no limitations with respect to the in-process inventory,
- there are less requirements with respect to the order in which tasks are executed,
- quality is difficult to measure (What is the quality of a decision?),
- quality of end-products may vary (in contrast to a product like a computer),
- transportation is timeless (at the speed of light), and
- production to stock is not possible (every product is unique).

Nevertheless, the two types of processes have a lot in common. Consider for example performance indicators such as throughput time, waiting time, service level and utilization. These performance indicators play a prominent part in both domains.

Many methods, techniques and tools have been developed to support the logistic control of manufacturing processes. MRP-I, MRP-II, BOM, OPT, JIT, TQM, EOQ, SIC, EPQ en DRP are some of the buzzwords used to identify the logistical principles successfully applied in this context (Buffa and Sarin [3]). Until now, the workflow-community (cf. WFMC [8]), involved in automating administrative processes, has neglected to properly use these logistical principles. Despite the differences between the two application domains, it is clear that the workflow-community could benefit from these logistical principles. Unfortunately, most WFMS vendors focus on the separation of processes and applications (i.e. pushing the control flow out of the applications), instead of the logistic control of the workflow. In this paper we try to utilize a specific logistic concept in the context of workflow modeling: the *bill-of-materials*. We will show that the bill-of-materials can be used to generate a process definition in terms of a *Petri net*. Petri nets are a well-known technique to model and analyze workflow processes (Ellis and Nutt [4], Van der Aalst [1]). On the one hand, Petri nets have a strong theoretical basis. On the other hand, Petri nets are close to the process modeling techniques used in today's WFMS's. Therefore, this paper constitutes a basis for the automatic configuration of a WFMS on the basis of a bill-of-materials.

This paper is organized as follows. In Section 2 we introduce the bill-of-materials in a workflow context. In Section 3 we discuss the use of Petri nets for the modeling of workflow processes. The relation between these two models is investigated in Section 4. Moreover, an algorithm is given to map a bill-of-materials onto a Petri net.

2 Modeling product structures using the bill-of-materials

The Bill-Of-Materials (BOM) is often used in manufacturing to capture the structure of the products to be produced (Orlicky [7], Buffa and Sarin [3]). A bill-of-materials specifies which materials are needed to manufacture a product. Consider for example the bill-of-materials shown in Figure 1. This bill-of-materials specifies that a car is assembled from an engine and a subassembly. The subassembly is assembled from a chassis and four wheels. Many production con-

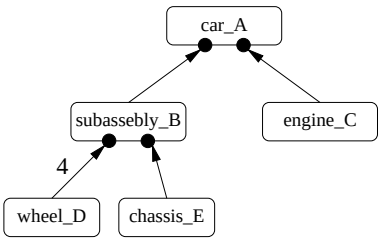

Figure 1: The bill-of-materials for a car.

trol systems are centered around the bill-of-materials. Material Requirements Planning (MRP-I) and Material Resources Planning (MRP-II) use the bill-of-materials as a starting point for the scheduling of the production process and the control of the inventory.

Administrative processes encountered in banking, insurance and government also produce products. The production of these workflow products corresponds to the processing of so-called *cases*. Examples of cases are tax declarations, traffic violations, insurance claims, purchase orders, licenses and loans. For these workflow products it is also possible to construct a bill-of-materials. Figure 2 shows the bill-of-

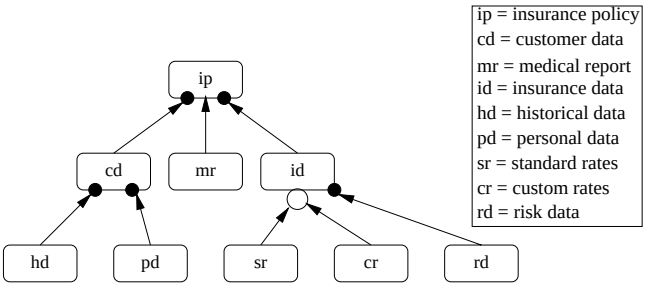

Figure 2: The bill-of-materials of an insurance policy.

materials of an insurance policy. An insurance policy consists of customer data (cd), insurance data (id) and possibly a medical report (depending on the type of insurance). The black dots indicate that the customer data and the insurance data are mandatory components. The customer data of an insurance policy consist of historical data (hd) and personal data (pd). The insurance data consist of risk data (rd) and information on either standard rates (sr) or customized rates (cr). The circle indicates that a choice is made between several components.

In contrast to the traditional bill-of-materials used in manufacturing, we assume that the quantity of each component used to assemble a product is equal to one. The tree-like representation shown in Figure 2 can be formalized as follows.

Definition 1 (Bill-of-materials) *A BOM is a tuple $(C, r, mandatory, optional, choice)$:*

- *C is a finite set of components,*
- *$r \in C$ is the root component,*
- *$mandatory \in C \rightarrow \mathbb{P}(C)$,*
- *$optional \in C \rightarrow \mathbb{P}(C)$,*
- *$choice \in C \rightarrow \mathbb{P}(\mathbb{P}(C))$,*

and satisfies the following properties:

- $\forall_{c \in C} \ |\{c' \in C \mid c \in mandatory(c')\}| + |\{c' \in C \mid c \in optional(c')\}| + |\{(c', cs) \in C \times C \mid cs \in choice(c') \ \wedge \ c \in cs\}| \leq 1$,
- *$R \subseteq C \times C$ such that $(c_1, c_2) \in R$ iff $c_1 \in mandatory(c_2) \cup optional(c_2) \cup \bigcup(choice(c_2))$,*
- *R represents a tree with root r, i.e., R is functional, acyclic and connected.*

The connections between the components C form a tree. The end-product (i.e. the processed case) is the root component r. Each component c has a number of mandatory components $mandatory(c)$ and optional components $optional(c)$. Moreover, for each $cs \in choice(c)$ precisely one component in cs is required to produce c. Each component appears only once in the bill-of-materials. Consider for example Figure 2: $mandatory(ip) = \{cd, id\}$, $optional(ip) = \{mr\}$ and $choice(id) = \{\{sr, cr\}\}$. A bill-of-materials specifies a relation R between components; $(c_1, c_2) \in R$ if there is an arrow from c_1 to c_2. To navigate through the bill-of-materials we introduce some additional notations.

Definition 2 *Given a BOM which specifies a relation R and a component $c \in C$, we define: $\hat{c} = \{x \in C \mid (c, x) \in R\}$, $\check{c} = \{x \in C \mid (x, c) \in R\}$, $\check{c} = mandatory(c) \cup optional(c) \cup choice(c)$. For $x \in \check{c}$ and $y \in \check{c}$: $\tilde{x} = y$ iff $x = y$ or $x \in y$.*

A bill-of-materials specifies a product structure. However, a WFMS requires a process definition to enact the workflow process. In this paper we use Petri nets for the specification of workflow processes.

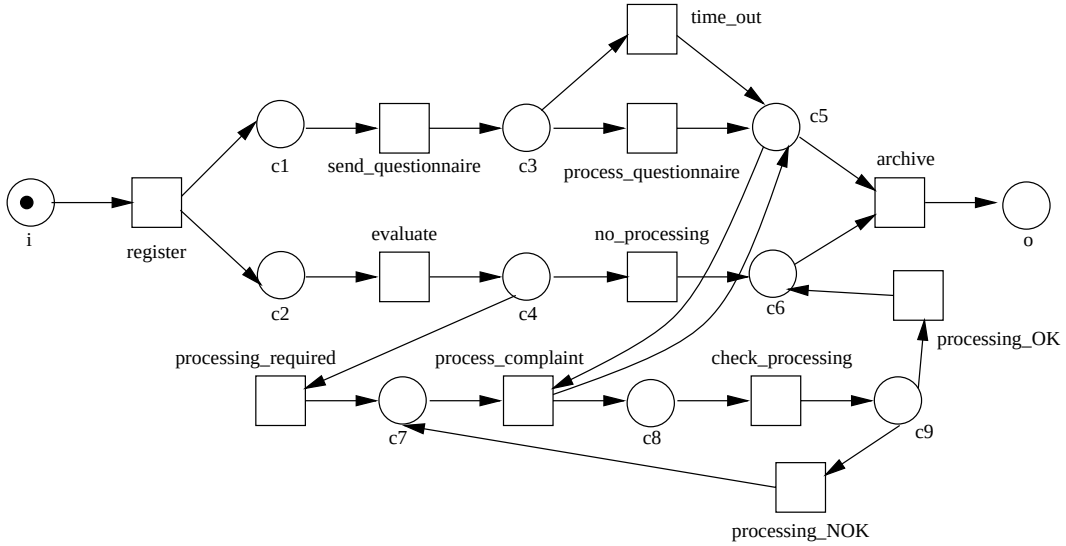

Figure 3: A Petri net for the processing of complaints.

3 Modeling workflow processes using Petri nets

The classical Petri net (Murata [6]) is a directed bipartite graph with two node types called *places* and *transitions*. The nodes are connected via directed *arcs*. Connections between two nodes of the same type are not allowed. Places are represented by circles and transitions by rectangles.

Definition 3 (Petri net) *A Petri net is a triple* (P, T, F)*:*

- P *is a finite set of places,*
- T *is a finite set of transitions* ($P \cap T = \emptyset$)*,*
- $F \subseteq (P \times T) \cup (T \times P)$ *is a set of arcs (flow relation)*

A place p is called an *input place* of a transition t iff there exists a directed arc from p to t. Place p is called an *output place* of transition t iff there exists a directed arc from t to p. At any time a place contains zero of more *tokens*, drawn as black dots. The *state*, often referred to as marking, is the distribution of tokens over places. The number of tokens may change during the execution of the net. Transitions are the active components in a Petri net: they change the state of the net according to the following *firing rule*:

(1) A transition t is said to be *enabled* iff each input place p of t contains at least one token.

(2) An enabled transition may *fire*. If transition t fires, then t *consumes* one token from each input place p of t and *produces* one token for each output place p of t.

Historically speaking, Petri nets originate from the early work of Carl Adam Petri. Since then the use and study of Petri nets has increased considerably. For a review of the history of Petri nets and an extensive bibliography the reader is referred to Murata [6].

A *workflow process* specifies which tasks need to be executed and in what order (Koulopoulos [5]). Such a process can be modeled by using building blocks such as the AND-split, AND-join, OR-split and OR-join. These building blocks are used to model sequential, conditional, parallel and iterative routing (WFMC [8]). Clearly, a Petri net can be used to specify the routing of cases. *Tasks* are modeled by transitions and causal dependencies are modeled by places. In fact, a place corresponds to a *condition* which can be used as pre- and/or post-conditions for tasks. An AND-split corresponds to a transition with two or more output places, and an AND-join corresponds to a transition with two or more input places. OR-splits/OR-joins correspond to places with multiple outgoing/ingoing arcs. Moreover, in [1] it is shown that the Petri net approach also allows for useful routing constructs absent in many WFMS's.

Figure 3 shows a Petri net which models the processing of complaints. First the complaint is registered (task *register*), then in parallel a questionnaire is sent to the complainant (task *send_questionnaire*) and the complaint is evaluated (task *evaluate*). If the complainant returns the questionnaire within two weeks, the task *process_questionnaire* is executed. If the questionnaire is not returned within two weeks, the result of the questionnaire is discarded (task *time_out*). Based on the result of the evaluation, the complaint is processed or not. The actual processing of the complaint (task *process_complaint*) is delayed until condition *c5* is satisfied, i.e., the questionnaire is processed or a time-out has occurred. The processing of the complaint is checked via task *check_processing*. Finally, task *archive* is executed. Note that sequential, conditional, parallel and iterative routing are present in this example.

4 Mapping the bill-of-materials onto Petri nets

4.1 Introduction

Figure 2 shows the bill-of-materials of an insurance policy. Figure 3 shows a Petri net which specifies the process of handling complaints. Clearly, these are two ways to view a workflow. The bill-of-materials provides a *product-centric* view and the Petri-net provides a *process-centric* view. Therefore, it is interesting to establish a relation between these two views. On the one hand it is useful to think in terms of the products that need to be 'manufactured', on the other hand it is important to focus on the process that needs to be controlled. Today's WFMS's are process centric, i.e., a process definition is needed to enact a workflow. Therefore, it is useful to be able to construct a Petri net based on the bill-of-materials.

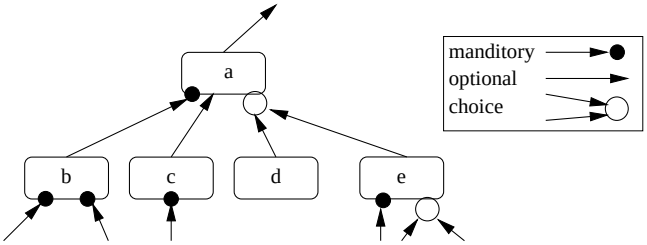

Figure 4: Some small piece of a bill-of-materials.

In this section we present an algorithm to construct a Petri net based on a product specification in terms of a bill-of-materials. For the algorithm it is assumed that there is a one-to-one relation between tasks and components, i.e., each component is produced by executing one task and each task corresponds to the production of one component. Figure 4 shows a bill-of-materials with a component *a* which is composed of a component *b*, a component *c* (optional) and either a *d* or an *e*. If we construct a Petri net for the bill-of-materials shown in Figure 4, then component *a* corresponds to a subnet responsible for the production of *a* and the components needed to produce *a*. The subnet starts with a transition $prepare_a$. This transition triggers the activities needed to produce *a*. Transition $prepare_a$ starts the production of *b*, *c* (optional) and either *d* or *e*. The choice between *d* and *e* is modeled by the place $in_{\{d,e\}}$, and the possibility to refrain from *c* is modeled by the by-pass via transition $skip_c$. The actual production of *a* is modeled by transition $produce_a$.

Component *d* in Figure 4 has no incoming arcs, i.e., no other components are needed to produce this product. Therefore, transition *d* corresponds to the production of *d*. The other components needed to produce *a* are *b*, *c* and *e*. These components require other components. This means that *b*, *c* and *e* need to be refined, i.e., each of these components corresponds to a network similar to the network constructed for *a*. The construction of the overall Petri net is an iterative procedure which starts with the root of the bill-of-materials and continues until all components have been considered.

Consider the bill-of-materials shown in Figure 2. By applying the iterative procedure we have just sketched, we obtain the Petri net shown in Figure 5.

4.2 Formalization

In Section 2 and 3 we have formally defined a bill-of-materials and a Petri net. Therefore, we can give an algorithm to construct a Petri net given a bill-of-materials. In this algorithm we use the navigation primitives defined in Definition 2.

Algorithm 1 *Let* $BOM = (C, r, mandatory, optional, choice)$ *be a bill-of-materials.*

Step 1
Construct the net $PN = (P, T, F)$ *with* $P = \{in_r, out_r\}$, $T = \{r\}$ *and* $F = \{(in_r, r), (r, out_r)\}$, *and* ***goto*** *step (2).*

Step 2
Use $PN = (P, T, F)$.

If $T \cap C = \emptyset$ ***then goto*** *step (4)* ***else select*** *a component* $c \in T \cap C$.

If $\check{c} = \emptyset$ ***then relabel*** *transition c to* $produce_c$ *and* ***goto*** *step (2)* ***else goto*** *step (3).*

Step 3
Replace transition c by a subnet which uses the following places and transitions:
$P_{out} := \{out_x \mid x \in \check{c}\}$, $P_{in} := \{in_x \mid x \in \check{c}\}$, *and* $T_{skip} := \{skip_x \mid x \in optional(c)\}$.

The modified net is defined as follows:
$P' := P \cup P_{out} \cup P_{in}$
$T' := (T \setminus \{c\}) \cup \{prepare_c, produce_c\} \cup \check{c} \cup T_{skip}$
$F' := (F \setminus \{(in_{\check{c}}, c), (c, out_{\check{c}})\}) \cup$
$\{(in_{\check{c}}, prepare_c), (produce_c, out_{\check{c}})\} \cup$
$\{(out_x, produce_c) \mid x \in \check{c}\} \cup \{(x, out_{\check{x}}) \mid x \in \check{c}\} \cup$
$\{(in_{\check{x}}, x) \mid x \in \check{c}\} \cup \{(prepare_c, in_x) \mid x \in \check{c}\} \cup$
$\{(skip_x, out_x) \mid x \in optional(c)\} \cup$
$\{(in_x, skip_x) \mid x \in optional(c)\}$

$PN := (P', T', F')$ *and* ***goto*** *step (2).*

Step 4
For each preparation transition (i.e. transitions of the form $prepare_x$*) with just one input place and one output place; remove the transition and fuse the input and the output place together.*

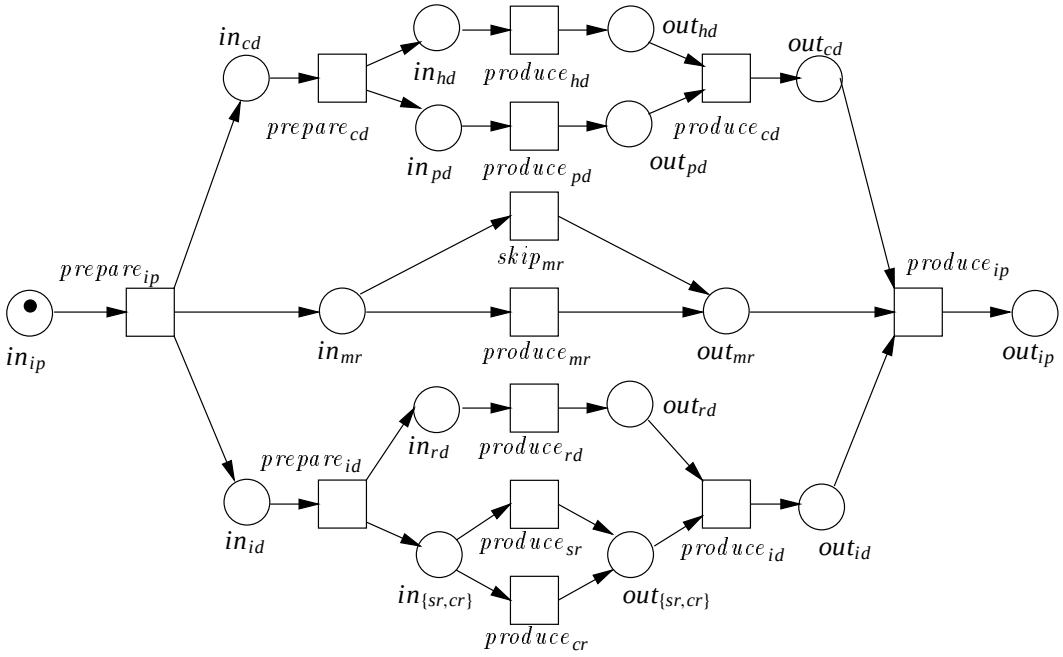

Figure 5: A Petri net which corresponds to the bill-of-materials shown in Figure 2.

4.3 Analysis of properties

The complexity of the workflows encountered in modern organizations is increasing. Therefore, we need methods and techniques to support both the modeling *and* analysis of these workflows. Petri nets often allow for a representation which is close to the intuition of the workflow designer. Moreover, the Petri net representation can be used as a starting point for various kinds of analysis. For an overview of the many analysis methods developed for Petri nets the reader is referred to Murata [6]. These methods can be used to prove properties (safety properties, invariance properties, deadlock, etc.) and to calculate performance measures (response times, waiting times, occupation rates, etc.). In this way it is possible to evaluate alternative workflows.

Based on the rich theory of Petri nets we can reason about the correctness of a workflow process. Therefore, it is interesting to see which properties are satisfied for any Petri net constructed via the algorithm presented in this section.
Let $BOM = (C, r, mandatory, optional, choice)$ be a bill-of-materials and let $PN = (P, T, F)$ be the Petri net constructed using the algorithm.

- PN is *safe* (1-bounded), i.e., for each case the maximal number of tokens in a place is equal to one. This means that the places correspond to conditions which are either true (place contains one token) or false (place is empty).
- PN is *(extended) free-choice*, i.e., if two transitions share an input place, then the sets of input places are identical. Free-choice nets have some very elegant properties and correspond to workflows where parallelism and choice are separated.
- If in_r is fused with out_r, then the resulting net is *strongly connected*. As a result, each task (transition) or condition (place) is on a path from in_r and out_r.
- If in_r is fused with out_r and this fused place is the only place containing a token, then the resulting net is *live*. This means that given a reachable state it is possible to fire any transition, i.e. all tasks can be executed.
- PN is *sound* ([2]), i.e., if we start in the state where in_r is the only place with a token (the initial state) then the following three properties hold:
  - For any reachable state it is possible to reach a state with a token in out_r.
  - The state which consists of just one token in out_r is the only reachable state with a token in out_r.
  - It is possible to fire each of the transitions at least once.

The soundness property is a very important property in the context of workflow management. A workflow process is sound if, for any case, the process terminates properly, i.e., termination is guaranteed, there are no dangling references, and deadlock and livelock are absent. In [2] several techniques are discussed to verify soundness. For free-choice Petri nets the soundness property can be verified in polynomial time. For arbitrary workflows represented in terms of a Petri net, soundness is decidable but also EXPSPACE-hard. Fortunately, for a Petri net constructed from the bill-of-materials, it is not necessary to use these techniques because soundness is guaranteed by the construction process itself.

5 Extensions

The Petri net shown in Figure 3 describes a workflow that cannot be constructed by using the algorithm introduced in the previous section. The workflow allows for iteration and testing of milestones. On the one hand, the task *process_complaint* may be executed several times (iteration). On the other hand, condition *c5* is a requirement for the execution this task (testing of milestones). Clearly, such a workflow process cannot be derived directly from a bill-of-materials such as the bill-of-materials shown in Figure 2. In practise we need workflow processes such as the process shown in Figure 3. There are two approaches to deal with these more advanced workflow processes and still use a bill-of-materials. First of all, it is possible to generate a default process based on the bill-of-materials by applying the algorithm. This default process is modified to incorporate additional routing constraints and tasks. Secondly, it is possible to furnish the bill-of-materials with additional information. To add this information, we need to extend the definition of a bill-of-materials. Some straightforward extensions are:

- *Precedence constraints.* Pieces of information can be used multiple times. To model this we need (additional) precedence constraints. The bill-of-material is an acyclic graph instead of a tree.
- *Grouping.* In general there is not a one-to-one correspondence between components in the bill-of-materials and tasks in the workflow process. This phenomenon can be modeled by grouping related components or tasks.
- *Iteration.* In general, iterations are undesirable. However, they are unavoidable if the result of a production step may be unsatisfactory. Therefore, we need to add this information to the bill-of-materials by indicating that certain components may require multiple production steps.

Many other extensions of the bill-of-materials can be added. For example, it is possible to reuse a bill-of-materials in another bill-of-materials (modular bill-of-materials). Concepts such as inheritance (generic bill-of-materials) and overriding (comparative bill-of-materials) can also be introduced.

6 Conclusion

In this paper we have presented an approach to (semi-)automatically generate a workflow process based on the product to be produced by the workflow system and its environment. Processes are represented in terms of Petri nets and workflow products are represented in terms of (extended) bills-of-materials. We have assumed that the process is generated on the basis of a bill-of-materials. This means that all the process requirements can be deduced from some kind of product-oriented description. In many situations this is not very realistic; both the product-centric view and the process-centric view are useful. Consider for example the distribution of work over the people involved in the processing of cases. This aspect is not addressed in the bill-of-materials, but is very important for the logistical control of the workflow process. Therefore, the process definition is often constructed from scratch without directly using the bill-of-materials. However, it is still useful to relate the bill-of-materials and the process definition. For each task in the workflow process we can specify the components that are created and/or used. Based on this information and the bill-of-materials it is possible to verify whether there are any conflicts between the causal order in the process and the bill-of-materials. Consider for example the Petri net shown in Figure 5. If we put $produce_{cd}$ and $produce_{pd}$ in parallel, then there is a conflict with the bill-of-materials in Figure 2 because the personal date (pd) are needed to produce the customer data (cd). In our opinion, the validation of the workflow process by comparing it with the bill-of-materials is an important topic for further research.

References

[1] W.M.P. van der Aalst. Three Good reasons for Using a Petri-net-based Workflow Management System. In S. Navathe and T. Wakayama, editors, *Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96)*, pages 179–201, Camebridge, Massachusetts, Nov 1996.

[2] W.M.P. van der Aalst. Verification of Workflow Nets. In P. Azema and G. Balbo, editors, *Application and Theory of Petri Nets 1997*, Lecture Notes in Computer Science, page (to appear). Springer-Verlag, Berlin, 1997.

[3] E.S. Buffa and R.K. Sarin. *Modern production/operations management.* Series in production/operations management. Wiley, 1987.

[4] C.A. Ellis and G.J. Nutt. Modelling and Enactment of Workflow Systems. In M. Ajmone Marsan, editor, *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, Berlin, 1993.

[5] T.M. Koulopoulos. *The Workflow Imperative*. Van Nostrand Reinhold, New York, 1995.

[6] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[7] J.A. Orlicky. Structuring the bill of materials for mrp. *Production and Inventory Management*, pages 19–42, Dec 1972.

[8] WFMC. Workflow Management Coalition Terminology and Glossary (WFMC-TC-1011). Technical report, Workflow Management Coalition, Brussels, 1996.