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

Workflow management promises a new solution to an age-old problem: controlling, monitoring, optimizing and supporting business processes. What is new about workflow management is the explicit representation of the business process logic which allows for computerized support. Business processes supported by a workflow management system are *case-driven* in the sense that tasks are executed for specific cases. A case corresponds to a service to the environment. Approving a loan, processing an insurance claim and handling a traffic violation are examples of case-driven processes. 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 can be used to describe the product that is manufactured using a workflow management system. This paper describes a technique to automatically generate a workflow process based on a bill-of-materials. It allows workflow designers to think in terms of the end-product instead of the internal process and constitutes a basis for the automatic configuration of a workflow management system on the basis of a bill-of-materials.

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

1 Introduction

From a logistical point of view, there are many similarities between administrative processes and production processes (cf. Platier [19]). 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:

1. *Making a copy is easy and cheap.*
   In contrast to making a copy of a product like a car, it is relatively easy to copy a piece of information (especially if it is in electronic form).
2. *There are no real limitations with respect to the in-process inventory.*
   Informational products do not require much space and are easy to access (especially

if they are stored in a database).

3. *There are less requirements with respect to the order in which tasks are executed.* Human resources are flexible and there are few technical constraints.

4. *Quality is difficult to measure.*
   What is the quality of the decision to accept an insurance claim?

5. *Quality of end-products may vary.*
   A manufacturer of cars has a minimal quality level that any product should satisfy. However, in an administrative process it might be attractive to skip certain checks to reduce the workload.

6. *Transportation of electronic data is timeless.*
   In a network information travels at the speed of light.

7. *Production to stock is seldom possible.*
   Every product is unique, therefore it is difficult to produce in advance. It is not possible to process an insurance claim before it arrives.

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 [9]). Until now, the workflow-community (cf. WFMC [20]), 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 (cf. Platier [19]). Unfortunately, most vendors of workflow management systems 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 workflow process definition. In this paper we specify workflow processes in terms of a *Petri net*. Petri nets are a well-known technique to model and analyze workflow processes (Ellis and Nutt [11], Van der Aalst [2, 4, 7]). Petri nets have a strong theoretical basis and are close to the process modeling techniques used in today's workflow management systems. Therefore, this paper constitutes a basis for the automatic configuration of a workflow management system on the basis of a bill-of-materials.

This paper is organized as follows. First, we discuss the relevance of workflow management (systems) and motivate the use of product structures in the context workflow management. In Section 3 we introduce the bill-of-materials in a workflow context. In Section 4 we discuss the use of Petri nets for the modeling of workflow processes. The relation between these two models is investigated in Section 5. Moreover, an algorithm is given to map a bill-of-materials onto a Petri net. This paper is an expanded version of an earlier paper [5].

2 Workflow management

In former times, information systems were designed to support the execution of individual tasks. Today's information systems need to support the business processes at hand. It no longer suffices to focus on just the tasks. The information system also needs to control, monitor and support the logistical aspects of a business process. In other words, the information system also has to manage the flow of work through the organization. Many organizations with complex business processes have identified the need for concepts, techniques, and tools to support the management of workflows. Based on this need the term *workflow management* was born.
Until recently there were no generic tools to support workflow management. As a result, parts of the business process were hard-coded in the applications. For example, an application to support task X triggers another application to support task Y. This means that one application knows about the existence of another application. This is undesirable, because every time the underlying business process is changed, applications need no be modified. Moreover, similar constructs need to be implemented in several applications and it is not possible to monitor and control the entire workflow. Therefore, several software vendors recognized the need for *workflow management systems*. A workflow management system (WFMS) is a generic software tool which allows for the definition, execution, registration and control of workflows. At the moment many vendors are offering a workflow management system. This shows that the software industry recognizes the potential of workflow management tools.

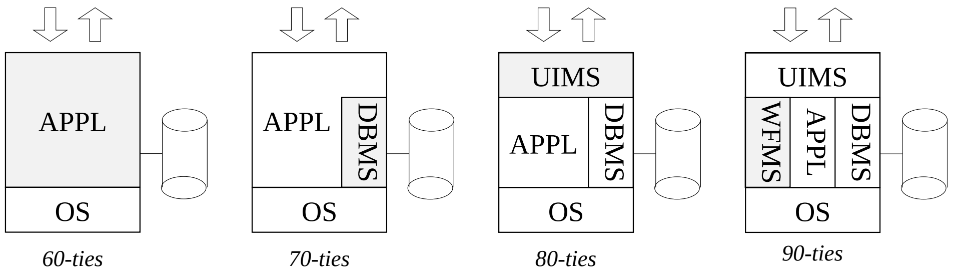

Figure 1: Workflow management systems in a historical perspective.

In order to become aware of the impact of workflow management in the near future, it is useful to consider the evolution of information systems over the last four decades. Figure 1 shows the phenomenon workflow management in a historical perspective. In the sixties an information system was composed of a number of stand-alone applications. For each of these applications an application-specific user interface and database system had to be developed, i.e. each application had its own routines for user interaction and data storage

and retrieval. In the seventies data was pushed out of the applications. For this purpose database management systems (DBMSs) were developed. By using a database management system, applications were freed from the burden of data management. In the eighties a similar thing happened for user interfaces. The emergence of user interface management systems (UIMSs) enabled application developers to push the user interaction out of the applications. In our opinion workflow management systems are the next step in pushing generic functionality out of the applications. The nineties will be marked by the emergence of workflow software, allowing application developers to push the business procedures out of the applications.
Figure 1 clearly shows that, in essence, the workflow management system is a generic building block to support business processes. Many information systems could benefit from such a building block, because many organizations are starting to see the need for advanced tools to support the design and execution of business processes. There are several reasons for the increased interest in business processes. First of all, management philosophies such as Business Process Reengineering (BPR) and Continuous Process Improvement (CPI) stimulated organizations to become more aware of the business processes. Secondly, today's organizations need to deliver a broad range of products and services. As a result the number of processes inside organizations has increased. Consider for example mortgages. A decade ago there were just a few types of mortgages, at the moment numerous types are available. Not only the number of products and services has increased, also the lifetime of products and services has decreased in the last three decades. As a result, today's businesses processes are also subject to frequent change! Moreover, the complexity of these processes increased considerably. All these changes in the environment of the information system in an average organization, have made business processes an important issue in the development of information systems. Therefore, there is a clear need for a building block named "workflow management system".

Clearly, today's workflow management systems focus on the support of workflow *processes*. However, the process is not a goal by itself! The output of the process is of the utmost importance. Therefore, it is important to define the output of the workflow process. One way to define the output is a specification of the end-product produced by the workflow process in terms of its components. In this paper we will show that it is possible to apply the bill-of-materials to the domain of workflow management. Moreover, we will show that it is possible to derive the workflow process on the basis of a bill-of-materials.

3 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 [17], Buffa and Sarin [9]). A bill-of-materials specifies which materials are needed to manufacture a product. Consider for example the bill-of-materials shown in Figure 2. 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 control systems are centered around the bill-of-materials. Material Re-

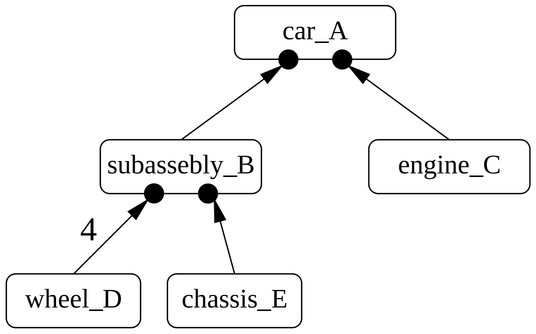

Figure 2: The bill-of-materials for a car.

quirements 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 3 shows the bill-of-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. Note that we extend the classical bill-of-materials with options and choices. In literature the term ‘variant bill-of-materials’ is used to denote such a product specification.

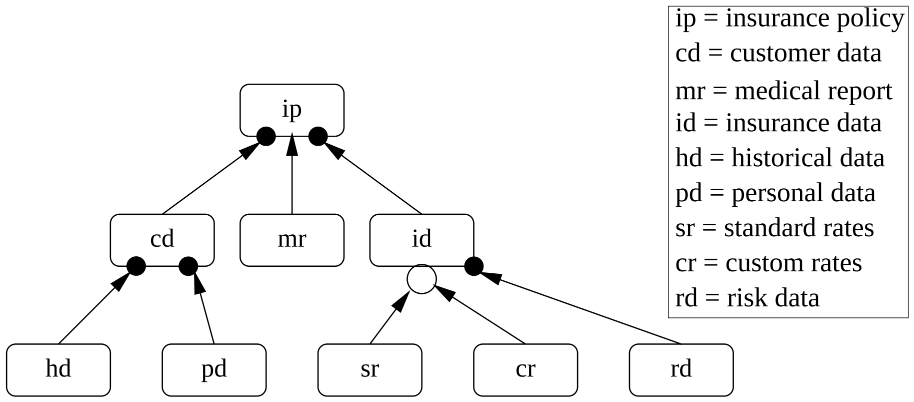

Figure 3: The bill-of-materials of an insurance policy.

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. Moreover, each component appears only once in the bill-of-materials. The tree-like representation shown in Figure 3 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 3: $mandatory(ip) = \{cd, id\}$, $optional(ip) = \{mr\}$ 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.

4 Modeling workflow processes using Petri nets

The main purpose of a workflow management system is the support of the definition, execution, registration and control of *processes*. Because processes are a dominant factor in workflow management, it is important to use an established framework for modeling and analyzing workflow processes [12, 14, 15]. In this paper we use a framework based on Petri nets. Petri nets are a well-founded process modeling technique. The classical Petri net was invented by Carl Adam Petri in the sixties ([18]). Since then Petri nets have been used to model and analyze all kinds of processes with applications ranging from protocols, hardware and embedded systems to flexible manufacturing systems, user interaction and business processes. In the last two decades the classical Petri net has been extended with color, time and hierarchy ([1, 13]). These extensions facilitate the modeling of complex processes where data and time are important factors. There are several reasons for using Petri nets for workflow modeling:

- *formal semantics*
  A workflow process specified in terms of a Petri net has a clear and precise definition, because the semantics of the classical Petri net and several enhancements (color, time, hierarchy) have been defined formally.

- *graphical nature*
  Petri nets are a graphical language. As a result Petri nets are intuitive and easy to learn. The graphical nature also supports the communication with end-users.

- *expressiveness*
  Petri nets support all the primitives needed to model a workflow process. All the routing constructs present in today's workflow management systems can be modeled. Moreover, the fact that states are represented explicitly, allows for the modeling of milestones and implicit choices.

- *properties*
  In the last three decades many people have investigated the basic properties of Petri nets. The firm mathematical foundation allows for the reasoning about these properties. As a result, there is a lot of common knowledge, in the form of books and articles, about this modeling technique.

- *analysis*
  Petri nets are marked by the availability of many analysis techniques. Clearly, this is a great asset in favor of the use of Petri nets for workflow modeling. These techniques 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 using standard Petri-net-based analysis tools.

- *vendor independent*
  Petri nets provide a tool-independent framework for modeling and analyzing processes. Petri nets are not based on a software package of a specific vendor and do

not cease to exist if a new version is released or when one vendor takes over another vendor.

The classical Petri net (Murata [16]) 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 squares.

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.

A *workflow process* specifies which tasks need to be executed and in what order (Koulopoulos [14]). 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 [20]). 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 [2, 4, 7] it is shown that the Petri net approach also allows for useful routing constructs absent in many WFMS's.

Figure 4 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

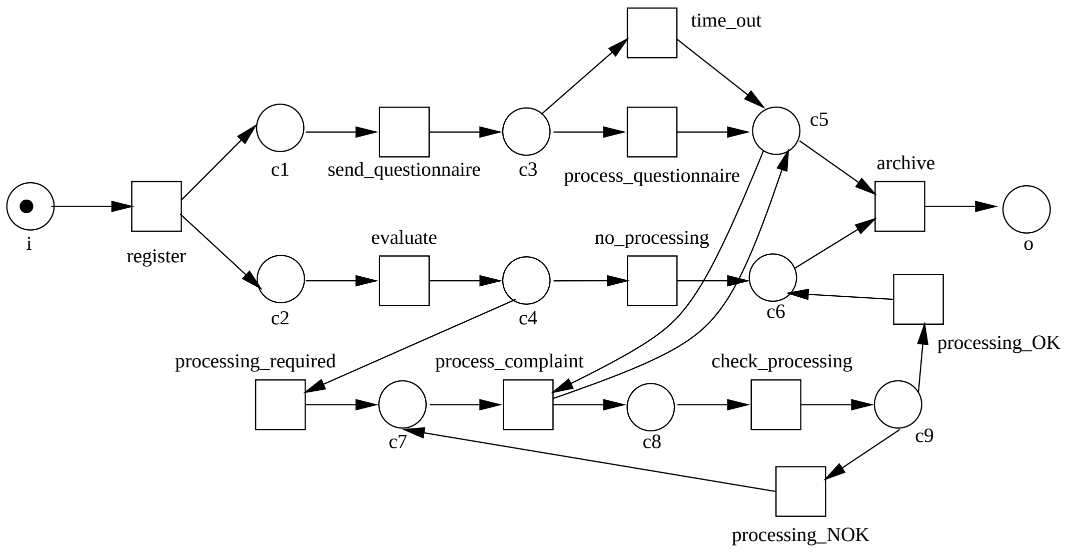

Figure 4: A Petri net for the processing of complaints.

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.

5 Mapping the bill-of-materials onto Petri nets

5.1 Introduction

Figure 3 shows the bill-of-materials of an insurance policy. Figure 4 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.

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 5 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

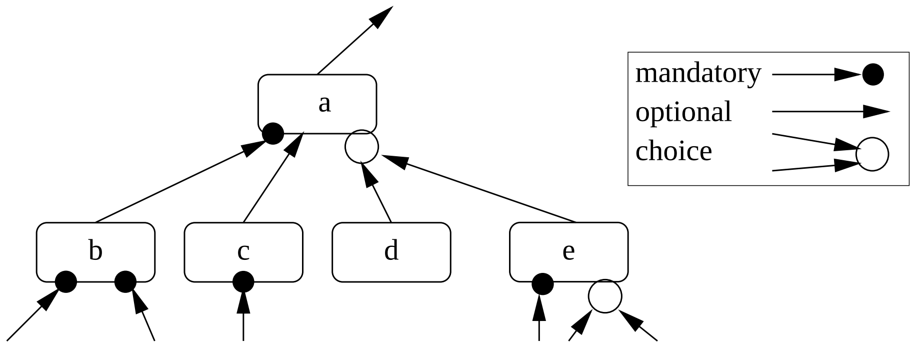

Figure 5: Some small piece of a bill-of-materials.

in Figure 5, 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 $produce_a$. Figure 6 shows the subnet which corresponds to the production of a.

Figure 6: The part of the constructed Petri net which corresponds to the production of component a in Figure 5.

Component d in Figure 5 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 d 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 3. By applying the iterative procedure we have just sketched, we obtain the Petri net shown in Figure 7.

Figure 7: A Petri net which corresponds to the bill-of-materials shown in Figure 3.

5.2 Formalization

In Section 3 and 4 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:
$P' := P \cup P_{out} \cup P_{in}$
$T' := (T \setminus \{c\}) \cup \{prepare_c, produce_c\} \cup \check{c} \cup T_{skip}$
$F' := (F \setminus \{(in_{\tilde{c}}, c), (c, out_{\tilde{c}})\}) \cup \{(in_{\tilde{c}}, prepare_c), (produce_c, out_{\tilde{c}})\} \cup$
$\{(out_x, produce_c) \mid x \in \check{c}\} \cup \{(x, out_{\tilde{x}}) \mid x \in \check{c}\} \cup \{(in_{\tilde{x}}, x) \mid x \in \check{c}\} \cup$
$\{(prepare_c, in_x) \mid x \in \check{c}\} \cup \{(skip_x, out_x) \mid x \in optional(c)\} \cup$
$\{(in_x, skip_x) \mid x \in optional(c)\}$

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

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

To illustrate this algorithm we use the bill-of-materials shown in Figure 8. By applying the algorithm we obtain the Petri-net shown in Figure 9.

5.3 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

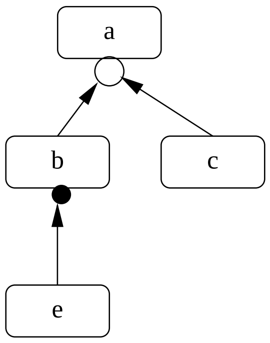

Figure 8: A bill-of-materials.

Figure 9: The construction of a workflow process definition based on Figure 8.

for various kinds of analysis. For an overview of the many analysis methods developed for Petri nets the reader is referred to Murata [16] and Desel and Esparza [10]. These methods can be used to prove properties (e.g. safety properties, invariance properties, deadlocks) and to calculate performance measures (response times, waiting times, occupation rates, etc.). In this way it is possible to evaluate alternative workflows.

The rich theory of Petri nets allows us reason about the correctness of a workflow process. Therefore, it is interesting to summarize some of properties that hold for any Petri net constructed by 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* ([10]), 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* ([3, 6]), 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 [6] 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.

6 Extensions

The Petri net shown in Figure 4 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. For example, the task *process_complaint* may be executed several times (iteration) and 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 because the resulting process will support advanced routing constructs suc as iteration. In practise we need workflow processes such as the process specified by the Petri net shown in Figure 4. 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. In this section, we briefly discuss some straightforward extensions.

6.1 Precedence constraints

One of the striking differences between administrative processes (office logistics) and manufacturing (production logistics), is the difference in the effort that is needed to produce an identical copy. Copying a physical component requires considerable effort. Therefore, the classical bill-of-materials forms a tree; every component is only used once. However, the effort to make a copy of a piece of information is negligible. Therefore, there is no reason why the same piece of information (component) should not be used twice. Using the same component multiple times, results in the introduction of (additional) precedence constraints. Consider for example the bill-of-materials shown in Figure 10. The personal data (pd) are used to produce the customer data. However, the personal data (pd) are also needed to produce the insurance data (id). This means that the production of the insurance data cannot start before the personal data are present. Therefore, we add a precedence constraint between the two components. Note that the precedence constraint is represented by a dashed arrow.

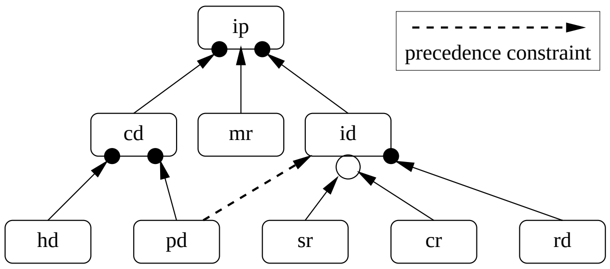

Figure 10: The bill-of-materials of an insurance policy with a precedence constraint.

Precedence constraints may introduce conflicts. Consider for example two components c_1 and c_2. If component c_1 is needed to produce component c_2 and c_2 is needed for c_1, then there is a conflict because of a cyclic dependency. Such a conflict corresponds to cycle in the bill-of-materials and causes deadlocks in the process definition. Therefore, we will not allow cycles. Moreover, it does not make sense to allow precedence constraints between two components if one of the components is needed and the other is not. Consider for example the bill-of-materials shown in Figure 10. A precedence constraint between mr and rd does not make any sense, because the medical report (mr) is optional. Based on these requirements we can formally define a precedence relation pre.

Definition 4 (Precedences) *Given a $BOM = (C, r, mandatory, optional, choice)$ the precedence constraints are modeled by a relation $pre \subseteq C \times C$ such that: (1) $R \cup pre$ is acyclic (R is the relation defined in Definition 1), and (2) if $(c_1, c_2) \in pre$, then there is a common ancestor reachable from both c_1 and c_2 via mandatory subset relations.*

Based on a bill-of-materials extended with precedence constraints, we can generate a Petri net. For this purpose we can use an algorithm similar to the algorithm introduced in the previous section. Every precedence constraint corresponds to a place $pre_{(x,y)}$ connecting two transitions $produce_x$ and $produce_y$. Figure 7 shows the Petri net which corresponds to the bill-of-materials shown in Figure 10.

Figure 11: The Petri net which corresponds to the bill-of-materials shown in Figure 10.

6.2 Grouping

Thus far, we have assumed that there is a one-to-one correspondence between components in the bill-of-materials and tasks in the workflow process. In general this is not true. The execution of one task may lead to the production of several products. This phenomenon can be modeled by grouping related components. In Figure 10 we can group hd, pd and cd by drawing a circle around this part of the bill-of-materials. This information can be used to replace the top four transitions in Figure 11 by a single transition which represents the production of hd, pd and cd. To avoid deadlocks the resulting workflow should not have any cycles.

It is also possible that the production of one component is spread over a number of sequential tasks. It is also possible to extend the bill-of-materials with this information.

6.3 Iteration

The workflow process constructed on the basis of a bill-of-materials does not allow for iteration, i.e. each task is executed only once. In general, iterations are undesirable. However, they are unavoidable if the result of production step may be unsatisfactory. If a task $produce_x$ can fail (i.e. the result is unsatisfactory), then the result of $produce_x$ is checked in another task $check_x$ and, depending on the result, $produce_x$ is executed again. This information can be added to the bill-of-materials by indicating that certain components may require multiple production steps. During the translation, the additional information can be used to introduce iteration in the workflow process by adding extra ”check tasks”.

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.

7 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. In this case it is also possible 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 7. If we put $produce_{cd}$ and $produce_{pd}$ in parallel, then there is a conflict with the bill-of-materials in Figure 3 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. Preliminary results show that the theory of the so-called implicit places (cf. Berthelot [8]) can be used to verify the consistency between the two complementary views on workflow management presented in this paper.

References

[1] W.M.P. van der Aalst. Putting Petri nets to work in industry. *Computers in Industry*, 25(1):45–54, 1994.

[2] W.M.P. van der Aalst. Petri-net-based Workflow Management Software. In A. Sheth, editor, *Proceedings of the NFS Workshop on Workflow and Process Automation in Information Systems*, pages 114–118, Athens, Georgia, May 1996.

[3] W.M.P. van der Aalst. Structural Characterizations of Sound Workflow Nets. Computing Science Reports 96/23, Eindhoven University of Technology, Eindhoven, 1996.

[4] 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.

[5] W.M.P. van der Aalst. Designing workflows based on product structures. In K. Li, editor, *Proceedings of the ninth IASTED International Conference on Parallel and Distributed Computing Systems*, 1997 (to appear).

[6] W.M.P. van der Aalst. Verification of Workflow Nets. In P. Azema and G. Balbo, editors, *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer-Verlag, Berlin, 1997.

[7] W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods and Systems (in Dutch)*. Academic Service, Schoonhoven, 1997.

[8] G. Berthelot. Transformations and decompositions of nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets 1986 Part I: Petri Nets, central models and their properties*, volume 254 of *Lecture Notes in Computer Science*, pages 360–376. Springer-Verlag, Berlin, 1987.

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

[10] J. Desel and J. Esparza. *Free choice Petri nets*, volume 40 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, Cambridge, 1995.

[11] 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.

[12] K. Hayes and K. Lavery. *Workflow management software: the business opportunity*. Ovum, 1991.

[13] K. Jensen. *Coloured Petri Nets. Basic concepts, analysis methods and practical use.* EATCS monographs on Theoretical Computer Science. Springer-Verlag, Berlin, 1992.

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

[15] P. Lawrence, editor. *Workflow Handbook 1997, Workflow Management Coalition*. John Wiley and Sons, New York, 1997.

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

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

[18] C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für instrumentelle Mathematik, Bonn, 1962.

[19] E.A.H. Platier. *A logistical view on business processes: BPR and WFM concepts (in Dutch)*. PhD thesis, Eindhoven University of Technology, Eindhoven, 1996.

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