PETRI-NET MODELLING OF AN ASSEMBLY PROCESS SYSTEM

D. GRADIŠAR [1,2] **and G. MUŠIČ** [2]

[1]*Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia.*
[2]*FE, Univ. of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia.*
E-mail: dejan.gradisar@ijs.si

Abstract: This paper describes how to apply timed Petri nets and existing production data to the modelling of production systems. Information about the structure of a production facility and about the products that can be produced is usually given in production-data management systems. We describe a method for using these data to algorithmically build a Petri-net model. The approach was implemented in the typical assembly process system where furniture fittings are produced.

Keywords: Timed Petri nets, Modelling, Production systems

1. INTRODUCTION

As the role played by information systems in production control increases, the need for a proper evaluation of the various decisions in both the design and operational stages of such systems is becoming more and more important.

In general, an appropriate model of a production process is needed in order to cope with its behaviour. However, this behaviour is often extremely complex. When the behaviour is described by a mathematical model, formal methods can be used, which usually improve the understanding of systems, allow their analysis and help in the implementation. Within the changing production environment the effectiveness of production modelling is, therefore, a prerequisite for the effective design and operation of manufacturing systems.

Petri nets (PN) represent a powerful graphical and mathematical modelling tool. The different abstraction levels of Petri-net models and their different interpretations make them especially usable for life-cycle design (Silva and Teruel, 1997). Many different extensions of classical Petri nets exist, and these are able to model a variety of real systems. In particular, timed Petri nets can be used to model and analyse a wide range of concurrent discrete-event systems (Zuberek, 1991; van der Aalst, 1995; Zuberek and Kubiak, 1999; Bowden, 2000). Several previous studies have addressed the timed-Petri-net-based analysis of discrete-event systems. (López-Mellado, 2002), for example, deals with the simulation of the deterministic timed Petri net for both timed places and timed transitions by using the firing-duration concept of time implementation. (van der Aalst, 1998) discusses the use of Petri nets in the context of workflow management. In (Gu and Bahri, 2002) the usage of Petri nets in the design and operation of a batch process is discussed. There is a lot of literature on the applicability of PNs in the modelling, analysis, synthesis and implementation of systems in the manufacturing-applications domain. A survey of the research area and a comprehensive bibliography can be found in (Zhou and Venkatesh, 1999). In (Recalde *et al.*, 2004) there is an example-driven tour on Petri

nets and manufacturing systems where the use of Petri-net production models through several phases of the design life-cycle is presented.

One of the central issues when using Petri nets in manufacturing is the systematic synthesis of Petri-net models for automated manufacturing systems. In (Huang *et al.*, 1995) a discrete-event matrix model of a FMS is used, which can be built based on standard manufacturing data, and can also be interpreted as a Petri net. This approach is particularly attractive when there are data about the production process available within some kind of production-management information system. Using these data, a model-building algorithm can be embedded within the information system. In this paper we propose a method for using the data from management systems, such as Manufacturing Resource Planning (MRP II) systems (Wortmann, 1995), to automate the procedure of building up the Petri-net model of a production system. Instead of using a discrete-event matrix model, the Petri net is built directly in a top-down manner, starting from the bill of materials (BOM) and the routings (Wortmann, 1995). The BOM and the routing data, together with the available resources, form the basic elements of the manufacturing process. These data can be effectively used to build up a detailed model of the production system with Petri nets. The product structure given in the form of the BOM and the process structure in the form of routings have also been used by other researchers. In (Czerwinski and Luh, 1994) a method for scheduling products that are related through the BOM is proposed using an improved Lagrangian Relaxation technique. An approach presented by (Yeh, 1997) maintains production data by using a bill of manufacture (BOMfr), which integrates the BOM and the routing data. Production data are then used to determine the production jobs that need to be completed in order to meet demands. Compared to previous work, the method proposed in this paper builds a Petri-net model that can be further analysed, simulated and used for scheduling purposes.

First we define timed Petri nets, where time is introduced by using the holding-durations concept. A general class of place/transition (P/T) nets supplemented by timed transitions is used. The practical experience also shows that for most of the applications in a real-manufacturing environment the use of deterministic time delays is sufficient. Adopting the class of timed P/T nets, a method for modelling the basic production activities with such a Petri net is described. A corresponding algorithm of automatic model building is presented. For a defined, timed Petri net a simulator was built, for which different heuristic rules can be introduced for scheduling purposes. The applicability of the proposed approach was illustrated using a practical scheduling problem, where the data about the production facility is given with the BOM and the routings. The model constructed using the proposed method was used to determine a schedule for the production operations.

In the next section we describe timed Petri nets that can be used for the modelling and analysis of a production system. In Section 3 the method for modelling the production system using data from the production-management system is presented. An illustrative application of modelling an assembly process and developing a schedule using timed Petri nets is given in Section 4. Finally, the conclusions are presented in Section 6.

2. TIMED PETRI NETS

Petri nets are a graphical and mathematical modelling tool that can be used to study systems that are characterised as being concurrent and asynchronous. The basic Place/Transition Petri net (Zhou and Venkatesh, 1999) is represented by the multiple:

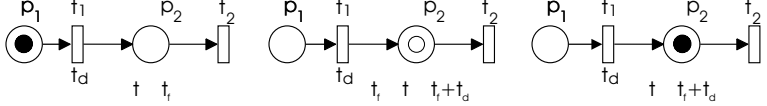

Fig. 1: Timed Petri net with holding durations.

$PN = (P, T, I, O, M_0)$, where:

- $P = \{p_1, p_2, ..., p_g\}$ is a finite set of places,
- $T = \{t_1, t_2, ..., t_h\}$ is a finite set of transitions,
- $I : (P \times T) \rightarrow \mathbb{N}$ is the input arc function. If there exists an arc with weight k connecting p_i to t_j, then $I(p_i, t_j) = k$, otherwise $I(p_i, t_j) = 0$.
- $O : (P \times T) \rightarrow \mathbb{N}$ is the output arc function. If there exists an arc with weight k connecting t_j to p_i, then $O(p_i, t_j) = k$, otherwise $O(p_i, t_j) = 0$.
- $M : P \rightarrow \mathbb{N}$ is the marking, M_0 is the initial marking.

Functions I and O define the weights of the directed arcs, which are represented by numbers placed along the arcs. In the case when the weight is 1, this marking is omitted, and in the case when the wight is 0, the arc is omitted. Let $\bullet t_j \subseteq P$ denote the set of places which are inputs to transition $t_j \in T$, i.e., there exists an arc from every $p_i \in \bullet t_j$ to t_j. A transition t_j is enabled by a given marking if, and only if, $M(p_i) \geqslant I(p_i, t_j)$, $\forall p_i \in \bullet t_j$. An enabled transition can fire, and as a result remove tokens from input places and create tokens in output places. If the transition t_j fires, then the new marking is given by $M'(p_i) = M(p_i) + O(p_i, t_j) - I(p_i, t_j)$, $\forall p_i \in P$.

An important concept in PNs is that of conflict. Two events are in conflict if either one of them can occur, but not both of them. Conflict occurs between transitions that are enabled by the same marking, where the firing of one transition disables the other transition. Also, parallel activities or concurrency can easily be expressed in terms of a PN. Two events are parallel if both events can occur in any order without conflicts. A situation where conflict and concurrency are mixed is called a confusion.

The concept of time is not explicitly given in the original definition of Petri nets. However, for the performance evaluation and scheduling problems of dynamic systems it is necessary to introduce time delays. Given that a transition represents an event, it is natural that time delays should be associated with transitions. As described in (Bowden, 2000), there are three basic ways of representing time in Petri nets: firing durations, holding durations and enabling durations. In this work timed Petri nets with deterministic time delays using holding-duration principle are used to model the behaviour of a production system. When using the holding-duration principle, a created token is considered unavailable for the time assigned to transition that created the token. The unavailable token cannot enable a transition and therefore causes a delay in the subsequent transition firings. This principle is graphically represented in Figure 1, where the available tokens are schematised with the corresponding number of undistinguishable (black) tokens and the unavailable tokens are indicated by the center not being filled. The time duration of each transition is given beside the transition, e.g., $f(t_1) = t_d$. When the time duration is 0 this denotation is omitted. In Figure 1, t denotes a model time represented by a global clock and t_f denotes the firing time of a transition.

Formal representation of the introduced timed Petri net is represented by the multiple:

$TPN = (P, T, I, O, s_0, f)$, where:

- P, T, I, O are the same as above,
- s_0 is the initial state of a timed Petri net.
- $f : T \rightarrow \mathbb{R}_0^+$ is the function that assigns a non-negative deterministic time-delay to every $t_j \in T$. The delays can be represented by a $1 \times h$ row vector $\mathbf{f}$ whose jth entry is $f(t_j)$.

The state of a timed Petri net is a combination of three functions $s = (m, n, r)$, where,

- $m : P \rightarrow \mathbb{N}$ is a marking function of available tokens. It defines a $g \times 1$ column vector $\mathbf{m}$ whose ith entry is $m(p_i)$.
- $n : P \rightarrow \mathbb{N}$ is a marking function of unavailable tokens. It defines a $g \times 1$ column vector $\mathbf{n}$ whose ith entry is $n(p_i)$.
- r is a remaining-holding-time function that assigns values to a number of local clocks that measure the remaining time for each unavailable token (if any) in a place. Assuming l unavailable tokens in p_i, i.e., $n(p_i) = l$, the remaining-holding-time function $r(p_i)$ defines a vector of l positive real numbers denoted by $\mathbf{r}(p_i) = [r(p_i)[1], r(p_i)[2], ..., r(p_i)[l]]$; r is empty for every p_i, where $n(p_i) = 0$.

A transition t_j is enabled by a given marking if, and only if, $m(p_i) \geqslant I(p_i, t_j)$, $\forall p_i \in \bullet t_j$. The firing of transitions is considered to be instantaneous. A new local clock is created for every newly created token and the initial value of the clock is determined by the delay of the transition that created the token. When no transition is enabled, the time of the global clock is incremented by the value of the smallest local clock. An unavailable token in a place where a local clock reaches zero becomes available and the clock is destroyed. The enabling condition is checked again.

3. THE MODELLING OF PRODUCTION SYSTEMS

This section deals with models for production facilities. These models play a role in the design and the operational control of a plant. Petri nets are a family of tools that provide a framework or working paradigm which can be used for many of the problems that appear during the life-cycle of a production system (Silva and Teruel, 1997). If used in all stages an additional benefit of improving the communication between these stages is achieved.

We present a method for modelling production systems using timed Petri nets based on data from production-management systems for the purpose of performance control. (van der Aalst, 1995) provides a method for mapping scheduling problems onto timed Petri nets, where the standard Petri-net theory can be used. To support the modelling of scheduling problems, he proposed a method to map tasks, resources and constraints onto a timed Petri net. In this paper a different representation of time in Petri nets is used, and the structure of the model is derived from existing production-management data (Wortmann, 1995).

3.1 The modelling of production activities

To make a product, a set of operations has to be performed. We can think of an operation as a set of events and activities. Using a timed PN, events are represented by transitions and activity is associated with the presence of a token in a place. First, a method of describing the production-system activities with timed Petri nets using the holding-duration representation of time is presented.

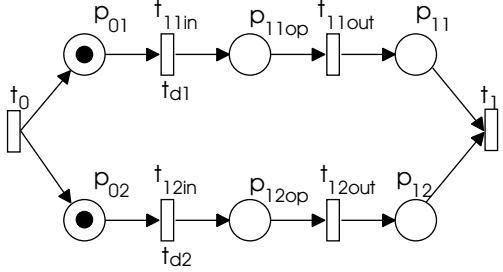

Fig. 2: Two parallel operations.

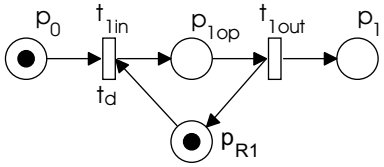

Fig. 3: Operation that uses a resource with finite capacity.

An elementary operation can be described with one place and two transitions, see Figure 1. When all the input conditions are met (there is a token in place p_1) the event that starts the operation occurs, t_1. This transition also determines the processing time of an operation. During that time the created token is unavailable in place p_2 and the operation is being executed. After that time the condition for ending the operation is being satisfied and t_2 can be fired.

When parallel activities need to be described the Petri-net structure presented in Figure 2 is used. The time delays of the transitions t_{11in} and t_{12in} define the duration of each operation. An available token in place p_{11out} (p_{12out}) indicates that operation is finished. Transition t_1 is used to synchronise both operations.

An operation might need resources, usually with a limited capacity, to be executed; this is illustrated in Figure 3. Place p_{R1} is used to model a resource. Its capacity is defined with the initial marking of that place. An additional place p_1 models the control flow. When the token is present in this place, subsequent operation can begin.

A particular operation can often be done on more different (sets of) resources with different availability, and the time duration can be different on each set of resources. An example where an operation can be executed on two different sets of resources is shown in Figure 4. If the operation chooses resource R_3, its time duration is determined with the transition $t_{2in} = t_{d2}$. Otherwise the set of resources, composed of R_1 and R_2, is being selected and its operation time is defined with $t_{1in} = t_{d1}$.

There are common situations where more operations use the same resource, e.g., an automated guided vehicle (AGV) in a manufacturing system or a mixing reactor in a batch system. This can be modelled as shown in Figure 5.

Precedence constraints are used to define technological limitations and the sequence of operations. An example of two successive operations is shown in Figure 6, depicted as $Op1$ and $Op2$. In this figure an example of technological limitations is also shown. Here, the situation where operation $Op1$ precedes operation $Op3$ is considered. For this purpose an additional place p_{pr1} is inserted between the transition t_{1out} (ending $Op1$) and the transition t_{3in} (starting $Op3$). The weight n of the arc, which connects p_{pr1} to t_{2in}, prescribes how many items need to

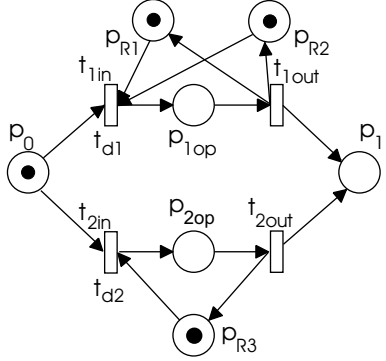

Fig. 4: Operation that can be done on two different sets of resources.

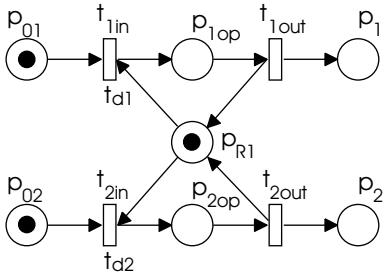

Fig. 5: Shared resource.

be produced by the first operation before the second operation can begin.

3.2 Modelling using the data from production-management systems

The most widely used production-management information system in practice is MRP II. Data stored in those systems can be used to build up a detailed model of the production system with Petri nets.

The BOM is a listing or description of the raw materials and items that make up a product, along with the required quantity of each. The BOM used in this work is defined as:

$BOM = (R, E, q, pre)$, where:

- $R = \{r_1\}$ is a root item.
- $E = \{e_1, ..., e_i\}$ is a finite set of sub-items,
- $q : E \rightarrow \mathbb{N}$ is the function that defines the quantities for each sub-item e_i. $\mathbf{q}$ represents an $i \times 1$ column vector whose ith entry is $q(e_i)$.

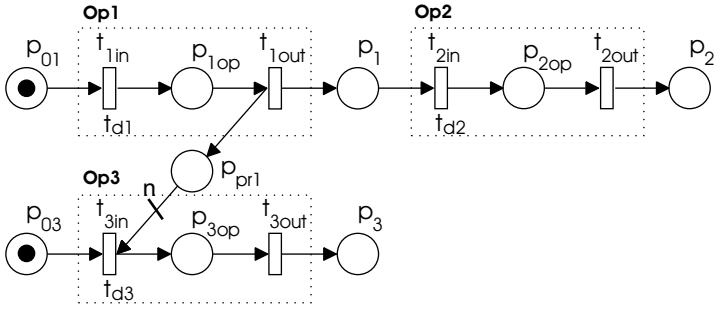

Fig. 6: Precedence constraint.

Table 1: Example of the BOM structure.

Item	Sub-item	Quantity	Precedence constraints
I	J	3	0 1
	K	2	0 0

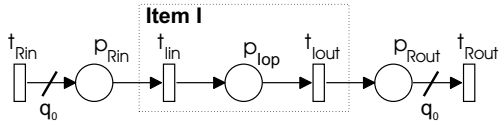

Fig. 7: PN structure representing one item in the BOM.

- $pre : (E \times E) \rightarrow \{0, 1\}$ is a precedence-constraints function. It defines the precedence-constraints matrix $\mathbf{pre}$, where $\mathbf{pre}(i, j) = 1$ indicates that the i-th item precedes the j-th item. It can also be interpreted as a directed graph.

R is a root item and represents the product that is composed of sub-items described with $e_i \in E$. The number of required sub-items is determined with the vector $\mathbf{q}$. When any sub-item has to be produced before another, the precedence graph pre is used to define it. All the sub-items have to be finished before the operation for the subsequent sub-items can begin. A required property of $\mathbf{pre}$ is that only zero values can be on its diagonal, i.e. a sub-item cannot precede itself. An item is never allowed to become (indirectly) a component of itself. In other words, if the BOM structure is seen as a directed graph, this graph should be cycle-free (Wortmann, 1995).

If any of the sub-items e_i are composed of any other sub-items, the same BOM definition is used to describe its dependencies. The items at the highest level of this structure represent a finished product, and those at the lower level represent raw materials. The items that represent raw materials do not have a BOM.

In Table 1 an example of a BOM describing the production of product I, which is composed of two components, i.e., three items of J and and two items of K. From the precedence-constraint matrix it is clear that all of the items J has to be completed before the production of item K can begin.

To start with let us assume that for each item from the BOM only one operation is needed. As stated before, each operation can be represented with one place and two transitions 1. To be able to prescribe how many of each item is required the transition t_{Rin} and the place p_{Rin} are added in front, and p_{Rout} and t_{Rout} are added behind this operation. The weight of the arcs that connect t_{Rin} with p_{Rin} and p_{Rout} with t_{Rout} are determined by the quantity q_0 of the required items. In this way an item I is represented with a Petri net as defined in Figure 7.

The finished product is defined with a structure of BOMs. In this way the construction of the overall Petri net is an iterative procedure that starts with the root of the BOM and continues until all the items have been considered. If the item requires any more sub-assemblies (i.e., items from a lower level) the operation, the framed area of the PN structure presented in Figure 7, is substituted with lower-level items. If there are more than one sub-items, they are given as parallel activities.

The substitution of an item with sub-items is defined as follows:

- Remove the place p_{Xop} and its input/output arcs.

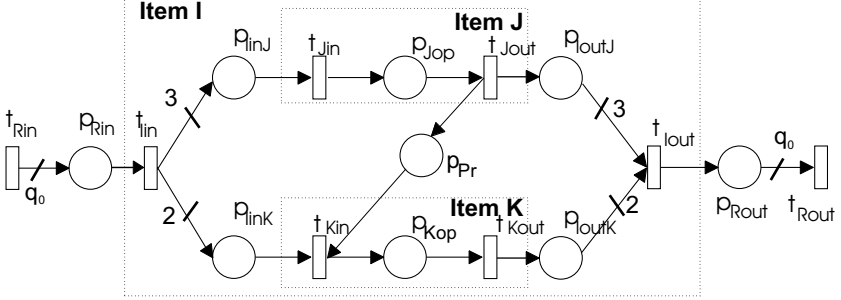

Fig. 8: BOM structure defined with PN.

Table 2: Routing of product K.

J	Operations	Duration	Resources
	Op10	10s/9s	R1/R2
	Op20	20s	R1, 3R3

- Define the PN structure for sub-components, as it is defined with a BOM: function *PN = placePN(R, E, q, pre)*. Consider the precedence constraints.
- Replace the removed place p_{Xop} by the sub-net defined in the previous step. The input and output transitions are merged with the existing ones: *PN = insertPN(PN, PN1)*. Structure PN1 is inserted to the main structure PN.

The result of building the PN model of this example (Table 1) is given in Figure 8.

For each item that can appear in the production process, and does not represent a raw material, a routing is defined. It defines a sequence of operations, each requiring processing by a particular machine for a certain processing time. The routing information is usually defined by a routing table. The table contains a header, where the item that is being composed is defined and the lines where all the required operations are described. For each operation one line is used.

As an example, the routing table for item K is presented in Table 2. Two operations are needed to produce this item. The first one can be done on two different resources, where the processing on each demands a different processing time. This is followed by the next operation, which needs two sets of resources: one resource of $R1$ and three of $R3$. A similar notation can be used for the other possible cases. Within the routing table concurrent operations are considered as a single operation with a parallel internal structure, as shown in Figure 2.

The implementation of the routing data in one component of a BOM is defined as follows:

- Remove the place p_{Xop} and its input/output arcs.
- Define a PN structure for the sub-components, as it is defined with routing data: function *PN1 = constructPN(PN, datRoute)*.
- Replace the removed place p_{Xop} by the sub-net defined in the previous step. The input and output transitions are merged with the existing ones: *PN = insertPN(PN, PN1)*. Structure PN1 is inserted to the main structure PN.

Each operation that appears in the routing is placed in the model using the function *constructPN()*. From the routing table the function yields the corresponding sequence of production operations and for each operation build a timed Petri net as defined in section 3.1. All

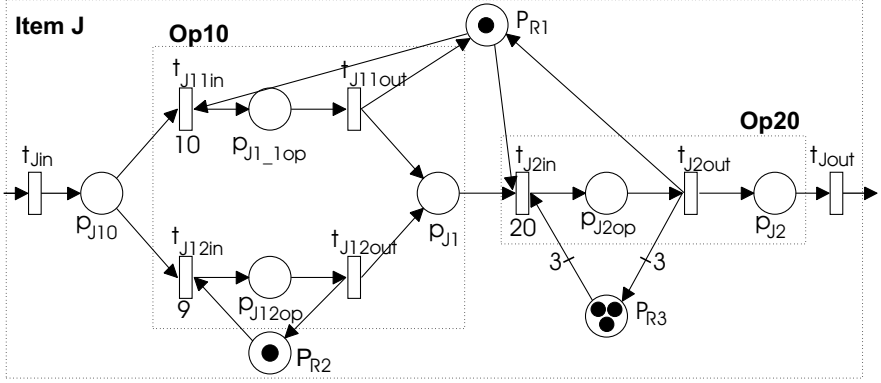

Fig. 9: Routing of product K modelled with timed Petri net.

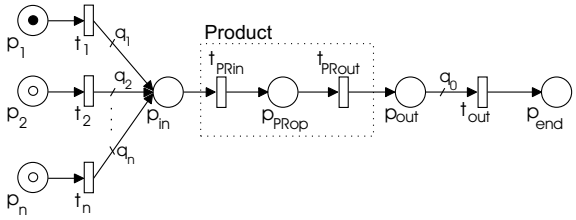

Fig. 10: Petri net structure of a work order.

the placed operations are connected as prescribed by the required technological sequence, and each operation is assigned to the required places representing appropriate resources. The PN structure in Figure 9 is achieved if the sequence of operations described with a routing table (Table 2) is modelled. The resulting PN structure is inserted into the main PN model, using the function *insertPN()*.

The routings are submodels that are inserted (by substitution, as defined previously) into the main model defined with the BOM structure. However, some activities of any sub-item may also be described with a BOM, i.e., in the case they are composed of semi-products. The construction of the overall Petri-net model can be achieved by combining all of the intermediate steps.

3.3 Procedure for building the PN model

The work order (WO) determines which and how many of the finished products have to be produced. Each product can be represented with a Petri-net model, shown in Figure 7, where one place is added in front and one at the end of the structure to determine the start and end of the work. The weight of the arc that connects t_{Rin} and p_{Rin} determines the number of required products. To be able to consider different starting times for different quantities of one product the general structure shown in Figure 10 is used. The clocks, which are assigned to the tokens that are in starting places, determine the starting time of every batch of products.

The modelling procedure can be summarised in the following algorithm:

Algorithm 1

[R, q, st] = *readWO*()
For i = 1 **to** *length*(R)
    E = *readBOM*(R(i))

```
    PN = placePN(R(i), E, q(i), [ ], st(i), x0, y0)
    PN = routing(PN, R(i))
end
```

First, the data about the WO are read. The products that are needed to be produced are given in R; in vector $\mathbf{q}$ the quantities of the desired products are passed; and vector $\mathbf{st}$ is used to determine the starting time of each product. For each product the Petri-net structure, shown in Figure 10, is determined and placed on the model. The step when the $routing()$ is called is described in more detail with algorithm 2:

Algorithm 2

```
function PN = routing(PN, R)
datRoute = readRouting(R)
[E, q, pre] = readBOM(R)
for i = 1 to length(datRoute.Op)
    if datRoute.Resources ==BOM
      PN1 = placePN(R, E, q, pre, [ ])
      PN = insertPN(PN, PN1)
      for j = 1 to length(E)
        PN1 = routing(PN1, E(j))
      end
    else
      PN = constructPN(PN, datRoute(i))
      PN = insertPN(PN, PN1)
    end
  end
```

First, the routing and the BOM data are read from the database. For each operation that comprises the routing, the algorithm checks whether it is made up of sub-item(s) or this is an operation. In the first case, the function *placePN()* is used to determine the PN structure of the given structure BOM. Precedence constraints are added if they exist. With the function *insertPN()* the resulting subnet is inserted into the main PN structure. If the operation represents the production operation, the function *constructPN*() is called. With it basic elements (Figures 1–6) are recognised, joined together and placed in the model, again using the function *insertPN()*. All the data about resources and time durations are acquired from the routing table. The described algorithm has been implemented in Matlab.

4. MODELLING OF AN ASSEMBLY PROCESS SYSTEM

The applicability of our approach will be demonstrated on a model of an assembly system where furniture fittings are produced. The production system is divided into a number of departments. The existing information-technology systems include a management system, which is used to plan the production and supervisory system, which is used to supervise the production process. To implement a detailed schedule, how the work should be done, an additional scheduling system should be implemented. The scheduling can be performed using timed Petri nets. The data needed to produce a Petri-net model can be retrieved from the existing information systems. In the presented model only a small part of the production process will be considered.

The process under consideration is an assembly process where different finished products are assembled from a number of sub-items. During the production process different production facilities are used to produce sub-items and finished products. The production facility considered in this example is shown in Figure 11. The process route of each finished product starts with a punching machine. The process continues through different resources, such as assembly lines, a paint chamber, and galvanisation lines. At the final stage there is another assembly line, where the finished products are assembled and packed.

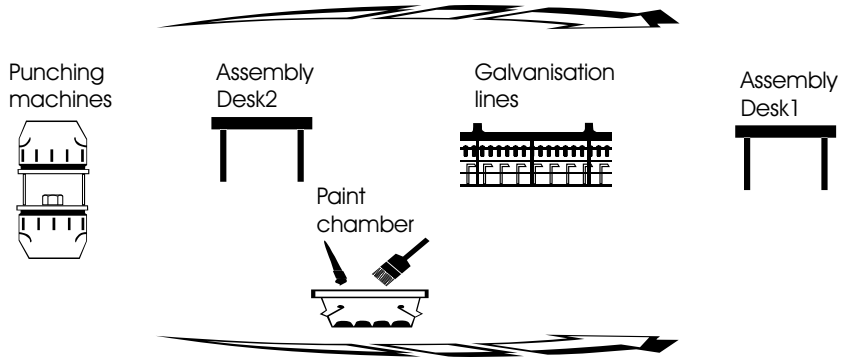

Fig. 11: Production plant.

In the considered problem four different types of products are produced: 'Corner clamp L', 'Corner clamp R', 'Clamp holder' and 'Angle-bar'. The request for which and how many products should be produced is given by work orders. Each work order also has its starting time. Table 3 lists the work orders considered in our case.

Table 3: Work orders for required products.

Product	Code	Quantity	Start time
Corner clamp L	CL	2	0
Corner clamp L	CL	2	80
Corner clamp R	CR	2	0
Clamp holder	CH	2	0
Angle-bar	AB	2	0

Each product is composed of one or more sub-assembly items. The structure of all products is described with the bill of material, Table 4. Where there are no precedence constraints between subitems, all the elements of the precedence-constraints matrix are 0, and the matrix is simply represented by $[0]$.

To build a particular item from the BOM list, some process steps are needed. These steps are described with routing data and are given in tables 5–7. The description of the data presented in the table will be given later during the modelling procedure.

The final product of one production stage represents the input material for the next stage. In this way the scheduling procedure should ensure that work orders are timely adjusted and in this way ensure that at each stage enough semi-products are produced. This kind of schedule is feasible. Using different heuristic rules it is possible to obtain a schedule that satisfies different objectives.

Table 4: Bill of materials.

Item	Sub-item	Quantity	Preced. constr.
BOM_CCL	AngBL (A)	1	
	Clamp (C)	1	
	Nut (N)	1	[0]
	Screw (S)	1	
BOM_CCR	AngBR (A)	1	
	Clamp (C)	1	
	Nut (N)	1	[0]
	Screw (S)	1	
BOM_ABL	AngB (B)	1	0 1
	Holder (H)	1	0 0
BOM_ABR	AngB (B)	1	0 1
	Holder (H)	1	0 0
BOM_HC	HolderC (HC)	1	
	Bracket (P)	1	[0]
	ScrewC (S)	1	

Table 5: Routings of each component from product 'Corner clamp L'.

	Operation	Duration	Resources
CL	Op1	–	BOM_CCL
	Op2	10	Desk1
AngBL	Op11	–	BOM_ABL
	Op12	10	Desk2
	Op13	20	Galvanisation1
Clamp	Op21	3	V1
	Op22	20	Galvanisation2
Nut	Op31	2	V1
	Op32	20	Galvanisation2
Screw	Op41	20	Galvanisation2
AngB	Op111	8	V1
Holder	Op221	8	V2

Table 6: Routings of each component from product 'Clamp holder'.

	Operation	Duration	Resources
CH	Op1	–	BOM_HC
	Op2	5	Desk1
HolderC	Op11	15	Galvanisation2
	Op12	10	Paint Chamber
Bracket	Op21	10	Paint Chamber
ScrewC	Op31	15	Galvanisation2

Table 7: Routings of each component from product 'Angle-bar'.

	Operation	Duration	Resources
AB	Op1	8	V2
	Op2	15	Galvanisation1
	Op3	5	Desk1

4.1 Modelling

Data from the BOM and the routings were used to build a Petri-net model. The development of the model will be presented for the part of the production where the left corner clamp is produced. As we can see from Table 3, there are two work orders: the first one has to start immediately with its production, while the starting time of the other is 80 time units later. When we apply the first step of our algorithm the PN structure, shown in Figure 12, is obtained. The production of different amounts of products starts at different times. Starting times are defined with the tokens in the starting places p_{WOCL1} and p_{WOCL2}, and the quantity of each is defined with the weights of the corresponding arcs.

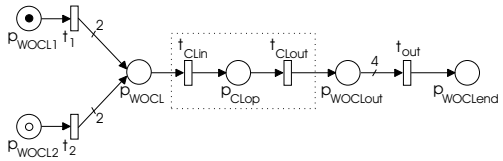

Fig. 12: Initial PN model of product 'Corner clamp L'.

From Table 5 the routing data about the product 'Corner Clamp L' (CL) are read. The first operation in this table ($Op1$) determines that the sub-items should be produced first as prescribed with 'BOM_CCL'. This BOM is defined in Table 4, and as we can see four different sub-items are needed ('Angle-bar with holder L', 'Clamp', 'Nut' and 'Screw'). It is clear from the table that for each item a designation character is given with its item-name. Those characters are used in the Petri-net model to indicate the item. When those subitems are produced, the production proceeds with an assembly – $Op2$. This situation is demonstrated with a Petri-net structure shown in Figure 13.

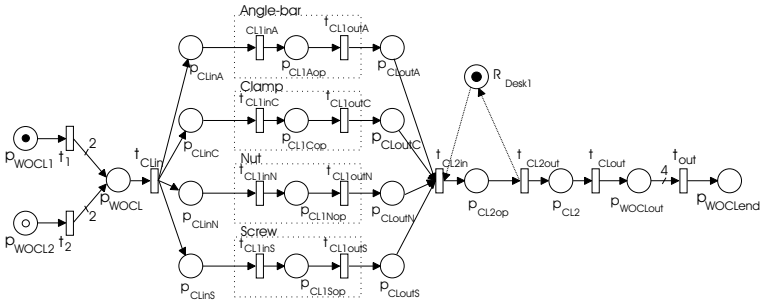

Fig. 13: 'Corner clamp L' is composed of four sub-items.

In the following the algorithm recognises the routing data for each of those sub-items. The routing data needed to build the sub-items of a 'Corner Clamp L' are given in Table 5. There is one operation needed to produce the sub-item 'Screw', two operations to produce 'Nut' and 'Clamp', and three operations to produce the 'Angle-bar with holder L'. Using these data the structure as defined in Figure 14 is achieved. Some denotations of the transitions and places

are omitted to achieve a clearer representation of the model. Also, the nodes that designate the start and end of production are not shown.

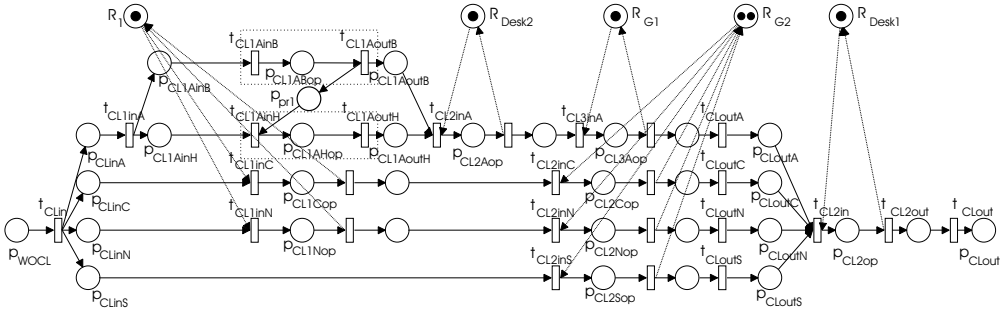

Fig. 14: Angle-bar with holder requires another two sub-items.

As we can see the production of angle-bar with holder ('AngBL') requires another two sub-items ('AngB' and 'Holder'). The precedence-constraint matrix defines that an item 'AngB' should be produced before 'Holder'. When all those details are included in the model we get the Petri-net structure presented in Figure 15.

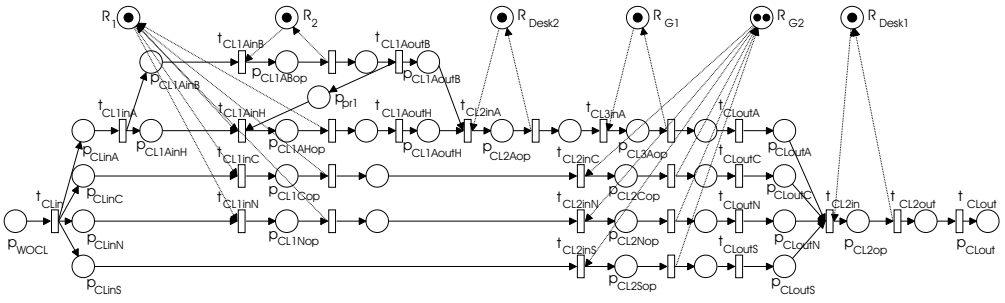

Fig. 15: PN model of product 'Corner clamp L'.

The same procedure was performed to model the production of all the other products, and the Petri-net model given in Figure 16 represents the given scheduling problem.

Finally, the resulting model is verified using P-invariant analysis. With this analysis we can determine 23 P-invariants. Seven of them are related to resources, two invariants refer to the precedence constraints and there are 14 invariants that result from every distinguishable product route. From this we can conclude that the resulting model reflects the system operation as defined by the MRP II system.

4.2 Results

The scheduling problem was mapped onto timed Petri nets, and the assembly process was modelled with timed Petri nets. We used Petri net simulator/scheduler was used to evaluate the different schedules of the tasks that are needed to produce the desired number of finished products. We tested different priority rules (SPT, LPT, etc.), and with them different schedules were achieved. The schedule allows an easy visualisation of the process and ensures that enough raw materials are available at the right time. It respects all the production constraints and the duration of the whole process can be identified from it. The shortest duration, i.e., 220 time units, would be achieved if the schedule was determined with the SPT priority rule, see Figure 17. In the other way, the schedule produced using LPT rule would take 225 time units.

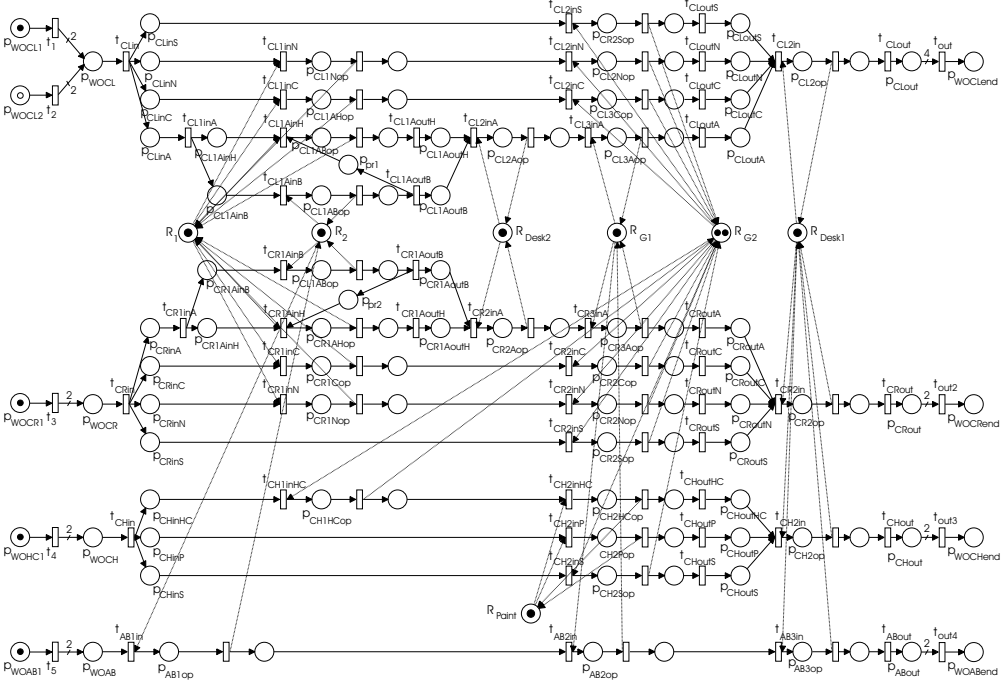

Fig. 16: PN model of all products.

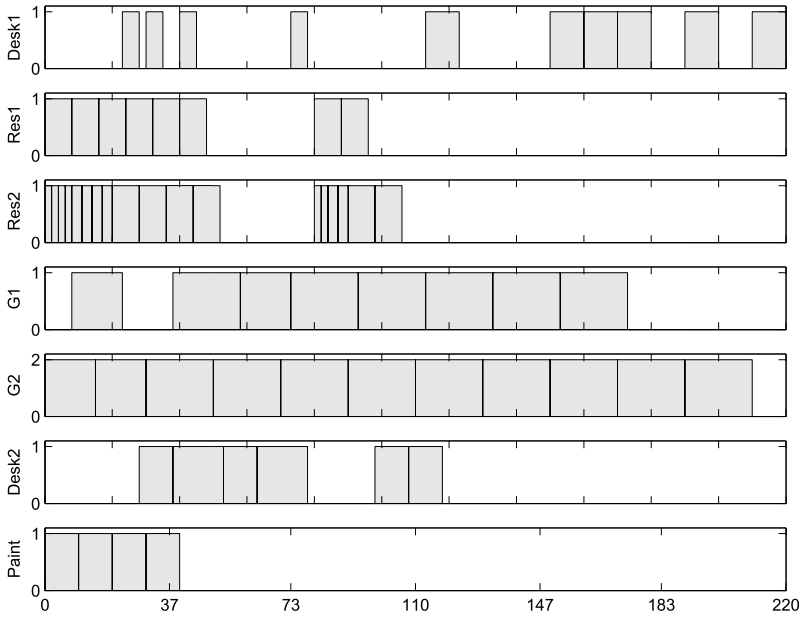

Fig. 17: Schedule of the tasks in a production process using the SPT priority rule.

The same problem was solved using the commercial scheduling tool Preactor. Also, in this way, the model of a production system was achieved using the data from the database of a management information system. Here the blocked-time approach is used (Enns, 1996) to schedule jobs. In this way all operations for a selected job (order) are scheduled, starting with the first operation and then forward in time to add later operations. The next job is then loaded until all the jobs have been done. In this way the schedule was achieved which would finish the job in 238 time units.

5. CONCLUSIONS

To be able to analyse a production system its mathematical model is required. Timed Petri nets represent a powerful mathematical formalism. In our work timed Petri nets with the holding-duration principle of time implementation were used to automate the modelling of a type of production system described by data from production-management systems. The production data are given with the BOM and the routings. The procedure for building a model using these data is presented. The applicability of the proposed approach was illustrated on an assembly process for assembly process. The model developed with the proposed method was used to determine a schedule for production operations. For the future work we plan to investigate the applicability of high-level Petri nets to the proposed modelling method.

REFERENCES

Bowden, F. D. J. (2000). A brief survey and synthesis of the roles of time in Petri nets. *Mathematical and Computer Modelling* **31**(10-12), 55–68.

Czerwinski, C.S. and P.B. Luh (1994). Scheduling products with bills of materials using an improved Lagrangean relaxation technique. *IEEE Transactions on Robotics and Automation* **10**(2), 99–111.

Enns, S.T. (1996). Finite capacity scheduling systems: performance issues and comparisons. *Computers and Industrial Engineering* **30**(4), 727–739.

Gu, T. and P. A. Bahri (2002). A survey of Petri net applications in batch processes. *Computers in Industry* **47**(1), 99–111.

Huang, H., F.L. Lewis, O.C. Pastravanu and A. Gurel (1995). Flowshop scheduling design in an FMS matrix framework. *Control Engineering Practice* **3**(4), 561–568.

López-Mellado, E. (2002). Analysis of discrete event systems by simulation of timed Petri net models. *Mathematics and Computers in Simulation* **61**(1), 53–59.

Recalde, L., M. Silva, J. Ezpeleta and E. Teruel (2004). Petri nets and manufacturing systems: An examples–driven tour. In: *Lectures on Concurrency and Petri Nets. Advances in Petri Nets* (G. Rozenberg J. Desel, W. Reisig, Ed.). Vol. 3098. pp. 742–788. Springer–Verlag.

Silva, M. and E. Teruel (1997). Petri nets for the design and operation of manufacturing systems. *European Journal of Control* **3**(3), 182–199.

van der Aalst, W. M. P. (1995). *Petri net based scheduling*. number 95. Eindhoven University of Technology Computing Science Reports/23.

van der Aalst, W. M. P. (1998). The application of Petri nets to workflow management. *Journal of Circuits, Systems and Computers* **8**(1), 21–66.

Wortmann, H. (1995). Comparison of information systems for engineer-to-order and make-to-stock situations. *Computers in Industry* **26**(3), 261–271.

Yeh, C.-H. (1997). Schedule based production. *International Journal of Production Economics* **51**(3), 235–242.

Zhou, M. C. and K. Venkatesh (1999). *Modeling, Simulation and Control of Flexible Manufacturing Systems - A Petri Net Approach*. World Scientific Publishing Co.

Zuberek, W. M. (1991). Timed Petri nets, definitions, properties, and applications. *Microelectronics and Reliability* **31**(4), 627–644.

Zuberek, W. M. and W. Kubiak (1999). Timed Petri nets in modeling and analysis of simple schedules for manufacturing cells. *Computers and Mathematics with Applications* **37**(11-12), 191–206.