# Modeling an Industrial Transportation Facility with Coloured Petri Nets

*Bernd Däne, Angela Mölders, Andrea Melber, Wolfgang Fengler*
*{bdaene;moelders;amelber;wfengler}@theoinf.tu-ilmenau.de*
*http://www.theoinf.tu-ilmenau.de/ra1/*

*Technical University of Ilmenau*
*P.O. Box 100565, D-98684 Ilmenau, Germany*

**Abstract:** The paper presents some results from a case study about the use of coloured Petri nets for modeling some aspects of production processes. Particularly it discusses a model of a complex transportation facility ("twin speed conveyor"). This example is explained, and some general results are derived. Methodological aspects such as integrating process, process control and process environment into one homogenous model are discussed. Results from analysis and simulation of the model are shown. The implementation of the model's process control partition into target software is discussed too.

**Keywords:** Petri net, process control, discrete model, simulation of manufacturing.

## 1 Introduction

This paper presents a case study for Petri Net based modeling of manufacturing processes. It describes some experiences concerning the creation and evaluation of models of transportation facilities. The use of coloured Petri Nets [Jensen92] leads to compact models of rather complex processes which include some repeating structures. To be useful for evaluation, the model includes stimulating and monitoring components which represent the process environment. The net's transitions can be attributed by at least duration and firing probabilities. In the evaluation process the model can be validated by formal analysis. Quantitative data such as statistical properties can be derived by simulation. The implementation of control software requires partitioning of the model to isolate the process control partition. The cut arcs at the border of this partition have to be replaced by input and output activities of the software. All these steps will be demonstrated by one of the developed models, the model of the so-called "twin speed conveyor".

## 2 Example: The "Twin Speed Conveyor"

The twin speed conveyor is a conveyor belt for transportation of discrete material such as boxes or cases. It has three parallel belts as shown in figure 1. The outer belts move at a low, the inner one at a higher speed. Since the inner belt normally lies below the others. In this case the boxes are moving at low speed during this time. But by activating a pusher, the inner belt is lifted so that one box moves at higher speed. A number of these pushers is distributed over the length of the conveyor and each can be activated individually.

The twin speed conveyor is used to transform a stream of randomly arriving boxes into a continuous stream, without bringing the boxes into direct touch. That means that the boxes arriving with random distances between each other shall leave the device with equal distances but variable velocity. This may

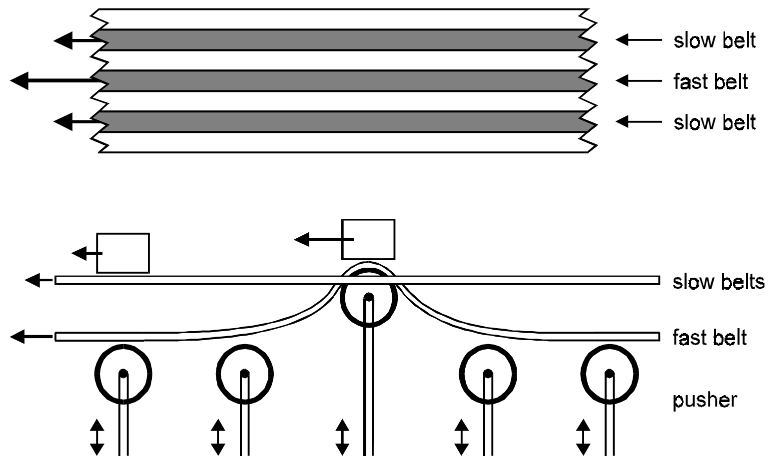

Figure 1: Belt configuration

be useful to adapt a facility that requires a continuous input stream such as a packaging device to a facility proving a disproportionate output.

The task cannot be done by queuing boxes with direct touch because they may be of a fragile kind. Instead,

the pushers shall be controlled in such a manner that the boxes move closer together. The outer belts' speed has to be varied to prevent "overflow" and "underflow", including "stop" if no box is available.

## 3 Modeling the Example Using Coloured Petri Nets

### Abstraction of the Process

First an abstract view of the problem is required, as abstract as possible and as accurate as needed. It's granularity is determined by the kind of information that is expected and has possibly to be changed during the evaluation.

At this point, a discrete time step is introduced so that the maximum output rate of the facility is one box per two clocks. The twin speed conveyor is partitioned into 23 positions. Each position has a length equal to *(length_of_box + minimal_distance)*. The positions are numbered as shown in figure 2. Each position except the last one is equipped with a pusher that forces the box to the next position when activated. This takes one clock, which means that the inner belt's speed is "one position per clock". The outer belts' speed can be "one position per 2, 4, 6, 9 or 14 clocks" or "stop" (these values are subject to change during optimization).

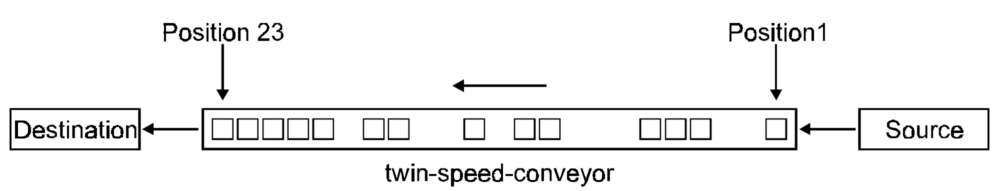

Figure 2: Position numbering

It is clear that this is only a time-discrete model approximation for a time-real system. In the model, the boxes do not move continuously but 'jump' from position to position. This approach has been taken to keep the model as simple as possible and to enable all possibilities of formal analysis. The question is the influence of this approximation to the desired results.

A much more precise approach would be the representation of the box positions by a continuous attribute. However, such an attribute implies less possibilities of formal methods for net analysis. A compromise would be a still discrete but finer grained step which would represent the box positions more exactly. The possibilities of analysis in this case are not restricted. But the complexity and the size of the model would increase significantly.

In the model presented in this paper we decided to remain at the coarse level. In reality the pushers act only at discrete positions and push the boxes exactly to the next position. This means that a pusher should be activated exactly when the next position is empty, i.e. the previous box has reached exactly it's new position. So the activation of the pushers does not need to be outside the chosen clock interval. Of course the movement of the outer belts is continuous, so that a normal moving box can be between two positions when the pusher is activated. In the model the box seems to stay at the 'old' position until it reaches the 'new' position. In reality, the distance of a movement initiated by pusher in this case would be shorter than one position, but both the final position after this movement and the activation time of the pusher match the chosen grid.

## Core Structure of the Petri Net Model

In the Petri Net model, each position of the conveyor is represented by a place which can be marked or not, referring to the presence or absence of a box at this position. So, the conveyor belt appears as a chain of places. The places will be connected by transitions to model the boxes' transit from one position to the next. Two groups of transitions are needed. One group represents the 'normal' movement and is activated simultaneously according to the outer belts' speed. The other transitions can be activated individually, thus representing the hurried movement of boxes when the appropriate pusher is activated.

This leads to a regular pattern with 23 places and 44 transitions which is partly shown in figure 3. The element names in figure 3 are chosen with respect to the later explained folded net. Places $p3_1$ to $p3_{23}$ represent the positions of the conveyor belt. Transitions $t4_1$ to $t4_{22}$ model the normal movement of the boxes. Their activation is controlled by inhibitor arcs. These inhibitor arcs will be connected to one common preplace, as will be shown later. In a similar manner, the transitions $t6_1$ to $t6_{22}$ represent the movement by pushers. They are controlled by inhibitor arcs too, but with individual activation.

The not very complex but extensive structure is folded into a compact representation by colouring. The place p3 (shown in figure 4) represents all 23 places while the distinction of the individual positions is managed by token colours. The token colours are named <1> to <23> according to the positions (for the token colours please refer to table 1 at the end of the paper). For the salce of clarity, these token colours have been used in figure 3, too.

Transition t4 stands for normal movement of boxes, t6 stands for activated pushers. Each of them represents 22 individual transitions. Their individual arc labels are represented by the symbols z1 and z2 (see table 3). For instance, if one prearc carries the label <14> the appropriate postarc wil have to carry the label <15>. So, the chain of places and transitions can be noted in a very compact manner.

The transitions t3 and t5 respectively stand for the 'input' and the 'output' of the conveyor. They have to be handled distinctively to t4 because they are connected to elements different from p3. Place p2 is needed

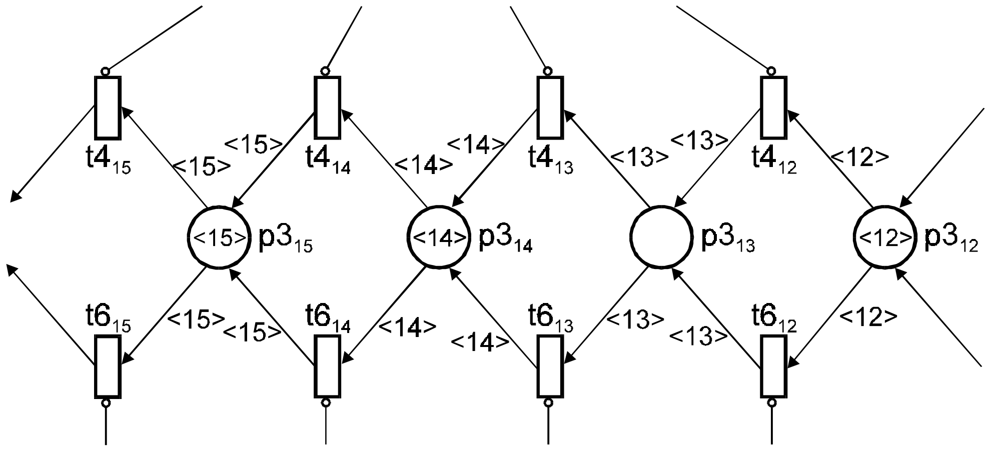


Figure 3: Verbose net structure (partly)

as 'input interface'. It deals as a buffer where the conveyor 'receives' new boxes. Please note that arcs without label refer to the colour <b> ('black') that is used when no further distinction is needed. The elements p2, p3, t3, t4, t5 and t6 establish the core of the model, e.g. the process itself.

## Extension of the Petri Net Model

This core model is extended by adding environmental aspects. This process environment is modeled by abstract behaviour only. In general, stimulation and observation components are needed. Only with the help of such components an evaluation of the model is possible.

For this model, a source and a destination of boxes are required. The destination is p9, where all boxes are collected. This place is needed for monitoring purposes.

The source 'produces' boxes in a random manner, similar to real conditions. The boxes arrive at random distances but at a given average rate. To observe the system's response to rapid changes of input rates, the stimulus randomly switches between *"average one box per 10 clocks"* and *"average one box per 3 clocks"*.

The elements p1, t1 and t2 do this job. Place p1 is responsible for the change of the average rate. It holds one token all the time, either in colour <r1> or <r2>. Colour <r1> refers to the lower rate, <r2> to the higher one. Firing of t1 switches the rates by 'recolouring' the token in p1 as shown with the help of symbols x1 and x2 (see table 2). In both cases the probability of firing p(t1) is 0.01 on average, which means that the rate is changed every 100 steps of simulation.

Transition t2 'produces' the new boxes. It's also equipped with a firing probability p(t2) according to table 2. The test arc establishes the relation to the token in p1. By choosing a probability of either 0.1 or 0.33 an average firing rate of either "once per 10 clocks" or "once per 3 clocks" is achieved.

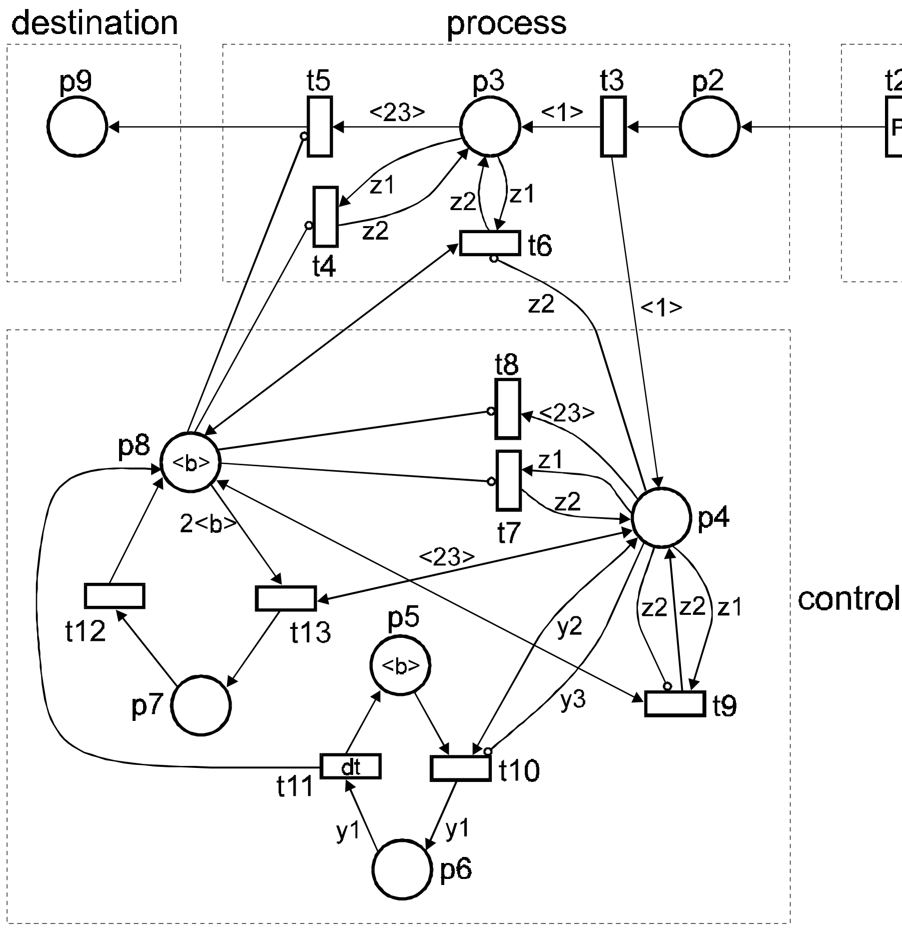


Figure 4: The complete net (see also the tables at the end of the paper)

**Definition of the Control Algorithm**

The next step is the design of a process control component. It has to observe the boxes in the process and to control the individual pushers and the speed of the outer belts. It is desired to keep the relation between the control partition and the process partition as lean as possible to prevent the generation of extensive interface structures in later steps. No connection is allowed between the control partition and the environmental components because these components do not allow any measurement or control access in the real system.

In this example the sensing facilities are minimized to one sensor indicating the arrival of one box at position 1. All other information about the whole process state can be derived inside the control partition by calculating all resulting changes in the process with respect to the generated control activities.

For this reason, a model of the process behaviour is needed inside the process controlling component. At this stage, this is a trivial task because this model already exists. Therefore, the process model is copied into the control component. Due to the uniform nature of the model concept no further conversion is required.

Place t4 is the copy of p3, and the transitions t7 to t9 are copies of t4 to t6. The attached arcs are copied, too. The arc from t3 to p4 represents the sensing facility that gives information about arriving boxes to the controlling component. It's clearly visible that the copy of the process will act synchronously to the process itself. This copy allows access to all necessary process state information.

The algorithm for the pusher activation is defined as *"pusher_i_on if box_at_position_i and not box_at_position_i+1"*. This is implemented by the inhibitor arc between p4 and t6 in conjunction with the already defined prearc of t6. The inhibitor arc disables the activation of the pusher when position i+1 is not empty. To keep the copy identical with the process, a similar arc between p4 and t9 is needed. Note that the sensing information in both cases comes from the process copy, not from the real process.

The outer belts' speed is controlled by the number of boxes in queue (without gap) starting at position 23. For instance, if three or four boxes stand in queue, this speed should be "one position per 9 clocks". Altogether, five different velocities are defined in this manner. If position 23 is empty the outer belts should stop because no box is available at the output.

The timing for the outer belts is produced by a state machine, consisting of p5, t10, p6 and t11. Transition t10 is responsible for the evaluation of the needed velocity. The different velocities will be coded by the colours <v1> to <v5> in p6 after the firing of t10. In table 5, the second row (starting with <v2>) refers to the example mentioned above. 'Three or four boxes in queue' means that at least the positions 21, 22 and 23 must be marked. This is sensed by the test arc between p4 and t10. In this case the value of y2 expands to *'<21>and<22>and<23>'*.

The case 'five boxes in queue' is attached to another velocity and refers to the next table row. To prevent conflicts between these cases, an additional inhibitor arc between p4 and t10 is needed. The symbol y3 in case of <v2> evaluates to *'<19>and<20>'*. Thus, this instance of the inhibitor arc is activated when more than one additional box (with respect to the three boxes sensed by y2) stand in queue.

In each situation only one instance of t10 is allowed to fire. The result is one token in p6 with colour <v1>, <v2>, <v3>, <v4> or <v5> meaning 'one position per 2, 4, 6, 9 or 14 clocks' (when position 23 is empty, t10 exceptionally does not fire at all). These velocities are selected respectively by 'at least 13, 7, 5, 3 or 1' boxes in queue. The more boxes are in queue, the higher the velocity will be. Thus, a controlling loop is established that controls the 'output rate' of the conveyor according to the average 'input rate' with suppression of rapid changes.

Transition t11 is equipped with the duration time dt(t11) according to table 5. It's firing takes as many clocks as table 5 shows according to the token colour in p6. Summarized with one clock for t10's firing the state machine generates the appropriate clock rates for the motion of the outer belts. Each time transition t11 completes its firing the outer belts have moved forwards one position.

Another state machine consists of p8, t13, p7 and t12. It transforms the firing rate of t11 into control information for the transitions that represent the outer belts' movement. Each time transition t11 pushes one token into p8, which has two tokens now, the firing of t13 is enabled by these two tokens (please remember that non-labelled arcs are considered as <b>). In the next clock one token is returned to p8 by firing of t12. The test arc between p4 and t13 disables t13 if position 23 is empty, resulting in stopped outer belts (remember again that all state information about the process is derived from the process state copy in p4).

Place p8 controls the inhibitor arcs of t5 and all instances of t4 which represent the outer belts' movement. Each of the cycles mentioned above leaves p8 empty for exactly one clock. In this clock the mentioned transitions are enabled and they complete the movement. Simultaneously, the pushers are deactivated by the test arc between p8 and t6. This prevents conflicts between t6 and t4 but does not influence the real system's behaviour because the firing result would be the same in this moment. Again, the copy of the process behaviour is kept identically to the process itself by applying the introduced inhibitor and test arcs to t7, t8 and t9 in a similar manner.

As shown in figure 4, the complex control algorithm could be represented by a not very extensive Petri net structure, mainly due to the use of token colours. In our case study an unfolded version of the net has been created, too, and checked against the coloured version in figure 4. The behaviour exposed no differences.

## 4 Evaluating the Model

### Formal Analysis

Formal methods allow to get information about fundamental properties of the net. They can be performed by Petri Net analysis tools such as INA [Starke92].

Some kind of analysis is useful to discover bugs, design rule violations and inconsistencies in the model. For example, the places $p3_i$ and $p4_i$ must be 1-safe to ensure that each position of the conveyor holds only one box at one time.

Furthermore, the integrity of the whole flow of physical parts may be checked. In particular this means that the total number of boxes shall not change at any time. Otherwise, the model needs some correction. This kind of analysis may be done by extending the model temporarily with a "source of boxes"-place which holds a number of tokens that represent the (now limited) number of boxes to be introduced into the system. This place establishes an additional precondition to t2. Now an invariant analysis is performed. The places p9, p3 and p2 plus the new place have to be a place-invariant, as shown in figure 5 . This proves the desired property.

Conflicts in the control partition of the net represent incomplete specification, which means that additional design decisions are needed. It's important to say that conflicts are not the same as errors. They appear during the process of development and indicate the places where further work is to be done. So, the automatic conflict detection by formal analysis can act as a helpful tool in project management.

Finally, the control partition has to be free of conflicts. Otherwise, the control algorithm would be non-deterministic and therefore not ready for implementing. However, the other partitions may contain conflicts as well. In this case the real non-deterministic behaviour of the systems is modeled. Formal analysis checks all combinations of different behaviour against the rules stated. For simulation purposes, a preferably stochastical attribute is needed to solve such conflicts.

Another kind of model check is the detection of deadlocks. Deadlocks often indicate errors, for instance in the specification of communication procedures between different parts of the algorithm. Formal analysis unveils such situations even if they occur under very special circumstances only. In some other cases deadlocks are expected, for example in 'shut-down'-procedures.

By reachability analysis the absence of unwanted states can be proved. In the example given, the synchronisation of the real process and the "inner model" inside the control partition can be checked for all circumstances.

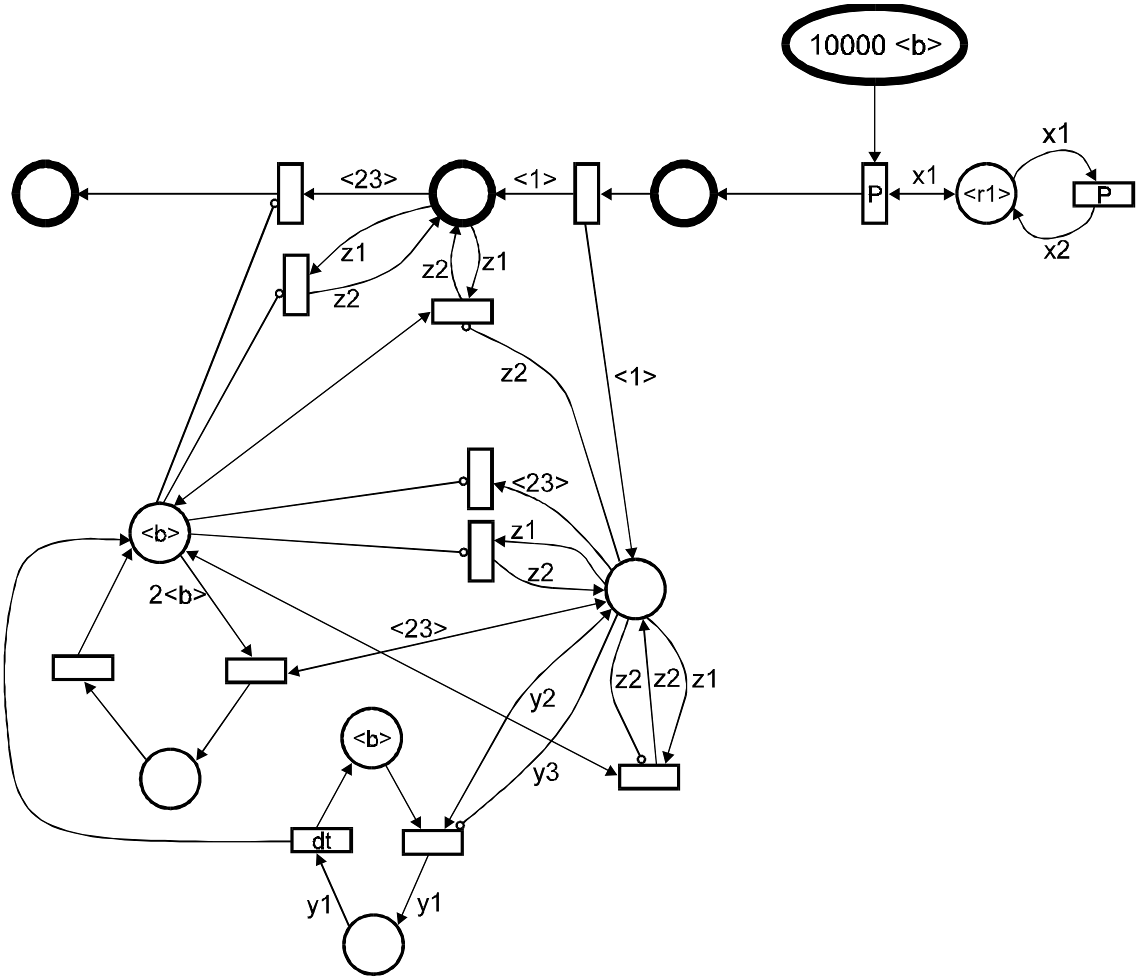


Figure 5: Invariant analysis. The emphasized elements represent a place-invariant

Furthermore, the model can be evaluated under worst case conditions, for instance with minimum and maximum input rates. For such purposes the stimulus partition may be manipulated. In the given example, the stochastically changing rate can be fixed to the minimum or the maximum. Reachability analysis can show, for instance, if the conveyor runs into ‘overflow’ (cannot accept incoming boxes) under such conditions.

**Simulation**

Another way to evaluate the model is simulation using animated Petri Net simulation tools. During simulation, the designer can observe the effects of changes in process and process environment conditions by simple manipulations in the model. The developer is enabled to experiment with variations in order to see the effects of particular changes in both the process and  the control partition. This work can be done without considerable turn-around-times because the simulation tool immediately interprets the net in it’s actual state.

Control algorithms may be changed rapidly without the need to write any line of code. If one is given an animation component, the results can be visualized in a clear and easy to understand way. Decisions show their consequences in an immediate manner. This intuitive approach, often underestimated by scientific research, leads to a faster process of development and makes it easier to find solutions for a problem and to avoid disadvantageous decisions.

In the given example, simulation was helpful when the fine-tuning of the slow belts' velocity control was done. In this step mistakes occurred that lead to malfunctions in already completed parts. Without simulation they surely wouldn't have been discovered immediately because other details were found by the developer. However, the clearly visible differences during simulation lead to rapid error recoverment.

Simulation can be used to predict quantitative properties of the process, especially statistical ones. For instance, the average number of boxes at the conveyor can be observed and recorded by a statistical evaluation component. Such measurements are valid with respect to the chosen stimulus parameters, which may be varied to get additional information.

For this project we used a tool for design and simulation of Petri Nets: "Peneca *CHROMOS*" [Wikarski96]. It has been developed at the Technical University of Ilmenau in corporation with OWIS Software GmbH, Martinroda. This tool includes an interface to the analysing tool INA [Starke92] and a component for statistical evaluation. This statistical component allows observation and recording of the net's firing behaviour as mentioned above.

## 5 Implementing Target Software

One goal of the modeling process is to give a basis for the implementation of target systems. This requires a clear splitting of the model into different partitions as shown in figure 4. The inside of the control partition has to be transformed into some kind of software. Inside the target software, a modified Petri Net interpreting algorithm may be implemented [Fengler91]. Another, generally faster but more costly approach is to transform the structure of the reachability graph into a programme structure.

Much attention has to be paid to the interface between the control partition and the rest of the model. Firstly, remember that the control partition shall be connected to the process partition only and not to the environmental parts of the model. The arcs cut at the border of the control partition have to be transformed into either input or output activities of the control software. In general, one must distinguish sensors and actors. Sensors pass information from the process to the control equipment, and actors influence the behaviour of the process when activated by the control equipment.

In the Petri net model the different kinds and directions of the cut arcs have to be considered in order to generate different kinds of interface constructs. Consequently, an isolated model of the control partition will be created. This model still appears in Petri Net manner but extended by input and output attributes [Fengler91] and will be ready for software generation.

From the view of the control partition, the following cases of cut arcs appear:

- *Incoming prearc:*
  Replace by an input activity that controls the firing of the associated transition in the control component. Create an extra output activity that is triggered when the transition fires.

- *Incoming test arc or inhibitor arc:*

Replace by an input activity that controls the firing of the associated transition in the control component.

- *Incoming postarc:*
  Replace by a new transition inside the control partition of which the firing is controlled by an input activity. Create an postarc to the associated place in the control component. If the considered postcondition is not safe create an extra output activity that statically depends on the token counts and colours in the associated place.

- *Outgoing prearc:*
  Replace by a new transition inside the control partition which is connected to the associated place by a prearc. Create an input activity that controls the firing of the new transition. Create an extra output activity that is triggered when this transition fires.

- *Outgoing test arc or inhibitor arc:*
  Replace by an output activity that statically depends on the token counts and colours in the associated place.

- *Outgoing postarc:*
  Replace by an output activity that is triggered when the associated transition fires. If the considered postcondition isn't safe then create an extra input activity that controls the firing of this transition.

In this design step manual interaction is needed more than in the others. In the twin speed example, the algorithm would produce an impulse-based output for controlling the speed of the outer belts. A better solution for real control equipment would be the use of the token colour in p6 as an output value to adjust the drive's speed. This can be done by defining an appropriate software activity manually which is associated to the place p6.

There are experimental tools that do the job in an (more or less) automated way. We have tested implementation components with target languages C/C++ and APL-2. The C/C++ implementation tool uses a special method named 'token flow analysis ' [Unger92]. It tries to derive an efficient programme structure from a given Petri net by evaluating the ways on which tokens can flow through the net. This method especially deals with approaches for efficient implementations on distributed systems.

At special request an implementation component with focus on the rather esoteric programming language APL/2 has been created and tested [Witter96]. Here the focus has been directed to powerful interfaces for input and output activities. The modular structure of APL/2 [Brown88] has been somehow useful for this task.

It is also possible to implement the control partition partly into hardware in a state machine manner. This may be necessary to deal with time-critical behaviour (probably not in direct process control but for tasks such as sensor interfacing or fieldbus protocol implementation [Nützel95]) or with high reliability demands ("emergency off" procedures for example) and leads to a contribution into hardware-software codesign methodology.

## 6 Conclusions and Further Work

This case study shows that Petri Nets can be used in all steps of the design process of industrial facilities. Their main advantage is the integration of the very different aspects and goals needed for the different design steps. It is also clear that human activities cannot be substituted at all, but the designer is disburdened from mechanical work and led to viewing systems instead of isolated parts. These advantages encourage the use of Petri Nets in design tools such as rapid prototyping environments.

Additional attributes in the net such as data structures or fuzzy extensions [Fengler96] increase modeling power but influence formal analysis in a negative way. They should be considered carefully before being introduced into a particular model. Another direction is the development of front-ends that hide the Petri Net by still graphical but more sophisticated and probably more application-specific layers [Mölders96].

## References


[Brown88] Brown, J. et al.: APL/2 at a glance.
Prentice Hall Inc. 1988

[Fengler91] Fengler, W.; Philippow, I.:
Entwurf industrieller Mikrocomputersysteme.
(Design of Industrial Microcomputer Systems. In German.)
Carl Hanser Verlag, München 1991

[Fengler96] Fengler, W.; Wendt, A.; Bogoljubow, J.; Däne, B.:
The Use of Fuzzy Coloured Petri Nets for Modeling and Simulation in Manufacturing.
ICATPN'96: Workshop for Manufacturing and Petri Nets, Osaka June 24-28, 1996

[Jensen92] Jensen, K.: Coloured Petri Nets. Springer Verlag. Berlin, New York. 1992

[Mölders96] Mölders, A.; Fengler, W.; Burkhardt, W.; Phillippow, I.:
Workflow Configuration Using the Object Process Model.
1st International Conference PAKM. Basel 1996

[Nützel95] Nützel, J.; Fengler, W. :
Using Formal Technics for Fieldbus Protocol Implementation.
11th ISPE/IEE/IFAC Conference on CAD/CAM, Pereira Aug. 28-30, 1995

[Starke92] Starke, P.H.: Integrated Net Analyser INA (Handbook).
Humboldt University Berlin, Department of Computer Science. 1992

[Unger92] Unger, H.; Ben Achour, K.:
A Petri Net Based Method to the Design of Parallel Programs for a Multiprocessor System.
In: Lecture Notes of Computer Science 634. Springer Verlag Berlin, New York. 1992

[Wikarski96] Wikarski, D. (Ed.):
Petri Net Tools: a Comparative Study (in English)
ISST-Bericht 39/96 of the Fraunhofer-Gesellschaft e.G., Berlin, Dec. 1996

[Witter96] Witter, M.:
Übersetzungskomponente für ein Petri-Netz-Tool mit der Ziel-Programmiersprache APL/2.
(Translation Component for a Petri Net Tool with Target Language APL/2. In German.)
Diploma Thesis 200-95D-054. TU Ilmenau 1996

## Tables

| Colour name | Meaning | Places |
|---|---|---|
| <v1>, <v2>, ... , <v5> | different speeds of slow belts | p6 |
| <r1>, <r2> | different stimulation rates | p1 |
| <1>, <2>, ... , <23> | correspond to conveyor positions | p3, p4 |
| <b> | others ('black') | p2, p5, p7, p8, p9 |

Table 1: Colours

| x1 | x2 | p(t1) | p(t2) |
|---|---|---|---|
| <r1> | <r2> | 0.01 | 0.1 |
| <r2> | <r1> | 0.01 | 0.33 |

Table 2: Colour table for t1 and t2

| z1 | z2 | |
|---|---|---|
| <i> | <i+1> | *i=[1,..,22]* |

Table 3: Colour table for t4, t6, t7, t9

| **Arcs without label: <b>** |
|---|
| **p:** probability of firing (normally 1.0)<br>**dt:** duration time (normally 1) |

Table 4: Attributes

| y1 | y2 | y3 | dt(t11) |
|---|---|---|---|
| <v1> | <23> | <21>∧<22> | 13 |
| <v2> | $\bigwedge_{i=21}^{23} \langle i \rangle$ | <19>∧<20> | 8 |
| <v3> | $\bigwedge_{i=19}^{23} \langle i \rangle$ | <17>∧<18> | 5 |
| <v4> | $\bigwedge_{i=17}^{23} \langle i \rangle$ | $\bigwedge_{i=11}^{16} \langle i \rangle$ | 3 |
| <v5> | $\bigwedge_{i=11}^{23} \langle i \rangle$ | - | 1 |

Table 5: Colour table for t10 and t11