Decision systems : the relation between problem specification and mathematical analysis

TECHNISCHE UNIVERSITEIT EINDHOVEN
Faculteit Wiskunde en Informatica

Memorandum COSOR 91-37

Decision systems; the relation between problem specification and mathematical analysis

J. Wessels

Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB Eindhoven
The Netherlands

Eindhoven, December 1991
The Netherlands
ISSN 0926-4493

Decision Systems the relation between problem specification and mathematical analysis

Jaap Wessels
*International Institute for Applied Systems Analysis
Laxenburg, Austria*

*Technical University Eindhoven
Eindhoven, The Netherlands*

Abstract

In this paper it is demonstrated that automated support for decision making of a tactical or strategic nature requires a solver-indepen
dent medium for describing decision situations. Such a medium may be specific for one environment, but it is also possible to develop media for certain types of environments. By using such a medium one obtains a decoupling of problem formulation and method of analysis. This makes it possible to use (parts of) the problem formulation as input for different types of models. Such problem formulations may provide mathematical models themselves, although they might also contain some less formal features.

The decoupling makes it possible to choose problem formulations which are much closer to the original decision situation than would otherwise be possible with formulations in terms of a preselected solver. The argumentation is illustrated by treating a language for specifying goods flow problems in some detail. This language is based on timed coloured Petri-nets.

1 Introduction.

The present paper is based on two types of experiences. In the first place the experience that only in exceptional cases the step from the decision situation to a mathematical model is a simple and natural one. A case where the step is relatively natural is the problem of routing and loading of trucks for the distribution of baking flour from the mill to the bakeries. Although, even in such a case, quite relevant constraints do not fit in the mathematical language. However, in most decision making problems the step from problem to model is large and, usually, it is also one of the most essential steps in the development of decision support systems. In the second place, we have experienced that only for operational decision making does one encounter the situation that a fixed sequence of steps leads from problem to model. Only in operational decision making one sees that essentially the same problem recurs, only with different data. In tactical and strategic decision making, however, one usually gets different types of problems subsequently, although they may use more or less the same data and other knowledge about the functioning of the system.

Example: a distribution structure

In order to keep a distribution structure in good shape in a changing world, one needs frequent adaptations. That means that not only operational decisions have to be made, but also tactical and strategic questions arise frequently and because of the complexity of these questions it would be good to have some tools for supporting this tactical and strategical decision making. However, the requirements for these tools are to be developed for a loosely described set of questions. Only later the real questions and how they are formulated appear. So, a main characteristic of pre-fabricated tools should be their flexibility and adaptability with respect to the types of questions they can be used for. Let us list some of these questions regarding a distribution structure in order to get some idea about their diversity:

a. Do we still need a distribution structure with three levels in a uniting Europe with higher demands on speed and stronger price-competition?

b. Do we produce all our products in each factory or will the distribution structure still be able to satisfy the performance requirements if the less demanded products are only manufactured by one factory?

c. If the factories specialize, is it then still necessary that each central warehouse stores all products or is it sufficient to forward the products only to the closest central warehouse (possibly located on the same premises as the factory)?

d. Do we still need so many regional warehouses?

e. Are the regional warehouses located in the right areas?

f. From where does a warehouse get its products? Would it be better to keep fixed rules for this allocation or replace them by dynamic ones?

g. How is the transport organised (frequent combined shippings, less frequent direct shippings or using an outside transport company)?

h. How do we fit a new line of products into the system?

i. How do we adapt the distribution structure after the merging with a regional competitor who possesses his own distribution system?

etc., etc.

This example clearly shows how the same data and the same mechanisms with regard to production, distribution and demand form the basis for answering a great variety of questions. Most of these questions cannot be answered directly with the given data and mechanisms, but require some form of advanced modelling and model analysis. For different questions one will need different types of models and different modes of analysis. For the location of facilities or allocation of production, linear programming might be useful. However, for allocation of regional warehouses to central warehouses we need some form of integer programming. For determining order levels, inventory theory will be useful and for evaluating the time performance of (parts of) the system, a queueing model might be sensible. For detailed checking of proposed new structures, it will be necessary to use simulation and scheduling models come in to mimic the daily operations. In fact, different questions of the types listed above will require non-standard models.

If one wants to make a decision support environment for the type of situation as described before, then the main arguments are *time* and *money*. "Time", since without a decision support environment each new question would require a time consuming analysis which easily takes more time than available. "Money", because such complete decision analyses are expensive, they not only cost much time, but also many man-hours. So the main reasons for investing in a decision support environment for tactical and strategic decision making are the speeding-up of analyses and the abatement of costs. Note that it is practically never the goal to make tools in such a way that the decision makers can do the analysis themselves. A reasonable aim is that the tools are such that analyses can be performed by staff members of the decision makers with only incidental help of computer programmers or analysts. Also for these seemingly modest aims it is not easy to design a decision support environment as will be clear from the list of components of such an environment:

a. A library of flexible algorithms for standard models.

b. Tools for making new algorithms for non-standard models.

c. A way of storing the knowledge about the situation. This knowledge consists of: data, physical procedures, control processes, and external influences.

d. Tools for translating this knowledge into models of a selected type.

e. Tools for specifying alternative systems and scenarios, which may regard any of the aforementioned features: data, physical procedures, information processes, control processes, external influences.

Essential for obtaining the possibility to use the same knowledge for different types of models is the *decoupling of problem specification and model formulation* and not only decoupling of model formulation and analytic tools. Note that *problem specification* entails on one hand the knowledge of the existing situation and possibly alternatives, but on the other hand, also scenarios for future behaviour if desirable.

It would be more efficient when the aforementioned components would be useful for more than one situation only. In fact, the chances for a library of algorithms to be useful in a wide variety of situations stand much better than for ways for storing knowledge about the situations. Such ways will usually be more specific. However, even for algorithmic libraries, a lot is still to be done in order to enhance the flexibility of programs and to obtain tools for making new algorithms for non-standard models.

The critical components, however, are the components mentioned under the labels c and e, which have the task of the specification of the current situation and possibly some alternatives. As said before, one cannot hope that this task can be performed by general-purpose tools. However, one might hope that specification tools can be developed for sizable classes of situations. Indeed, several attempts have been made, particularly for classes of highly technical decision problems (for flexible manufacturing problems, see Silva and Valette [21]; Van der Aalst and Waltmans [3]; for scheduling problems, see Carlier, Chretienne, Girault []; Hatono et. al. [9], Tamura, Hatono [22]; for cyclic job shops or assembly systems, see Harhalakis, Laftit, Proth [8]; for communication and computer systems, see Garg [7], Holiday, Venon [14], Magott [16], Molloy [19]).

The next four sections of this paper will be devoted to the introduction of an approach for specifying a broad class of goods flow situations. The exposition is not so much devoted to the technicalities, but rather emphasizes the possibilities and difficulties of such an approach. Section 2 states the problem and introduces the main tool, viz. Petri nets. In Section 3, time is added to increase the descriptive power. Several typical features for goods flow situations are described in this section. Section 4 is devoted to the use of hierarchical structures and parametrised modules. Section 5 introduces some ways of analysing specifications of the type introduced in previous sections. Finally, Section 6 contains some comments and conclusions.

2 Specifying goods flow situations.

In Sections 2-5, we will introduce an approach for specifying goods flow situations for use in decision support. The ideas for this approach were stimulated by a research project at the Technical University of Eindhoven in cooperation with and sponsored by the Dutch Organization for Applied Research TNO. The author uses some of the technical developments in this project to illustrate his views on decision support environments. The author is greatly indebted to the other members of the project team for the discussions which helped forming his view. In particular, his thanks go to Wil van der Aalst and Kees van Hee. The presentation relies strongly on the specification tool ExSpect which has been developed at the Technical University Eindhoven. However, these sections should not be considered as an exposition of this tool. For such an exposition the reader is referred to Van Hee et. al. [10] and to Van der Aalst [2]. In the present exposition ExSpect is only a vehicle for explaining the author's views on decision support environments. The existence of ExSpect also serves as a proof that these views are not completely unrealistic. The presentation has not as a goal to convince the reader that this is the right approach. A lot of further development and testing will be required before the aproach is really in a mature state. After all, the conclusion might even be that this is not the right way to proceed. In fact, the main danger is that the approach is too ambitious. Even so, the great attractiveness of the approach is that it heads right to the center of the crucial aspect of decision support, namely the specification of situations and alternatives. The main reason to present this research-in-progress here is that it shows that focussing on an approach for this crucial aspect gives a completely different view on the usual ways of treating decision support and also on the difficulties encountered with regard to the acceptance of such efforts.

The approach is ambitious in different ways. In the first place because of the large range of practical situations emphasized, namely virtually all goods flow situations where the different stages of the process are separated by discrete events. This would mean that practically all batch-type chemical production situations would be comprised as long as the highest level of detail is the batch. Of course, it is technically possible to go into more detail by splitting-up the processing of a batch into phases which are separated symbolically by discrete events. However, this would not lead to a natural specification of the goods flow. The second aspect where the ambition of the approach becomes clear is the fact that it aims at giving specifications which are very close to the real-life process and allowing the user to translate this specification into different types of models. In the usual modelling languages the starting point is a fixed type of analysis (for instance, linear programming or simulation) and the task of the language is to avoid computer programming technicalities (in the simulation case) and to simplify the modelling task itself (in both cases). A third ambition is to develop a specification tool which is so unambiguous that it can be understood as an executable prototype with which experiments can be performed. A fourth ambition is that it can be used for a large range of decision problems comprising the selection of the basic distribution structure on one hand and the control rules for a warehouse on the other. Finally, it is the ambition that the specification integrates the physical layer of the goods flow with the information flow and the process control.

A specification tool for goods flow problems has to be built on representations for the most elementary physical steps in the goods flow:

1. transformation of certain goods into others;
2. displacement of goods;
3. buffering of goods.

The elementary physical steps have a striking similarity with the elementary steps in information processing. If we combine this feature with the ambition to integrate *goods flow* and *information flow* into one specification, then it becomes obvious that approaches, which have shown some usefulness for specifying information flows on different levels, might provide a good starting point for our goal. Particularly, since *concurrency* and *parallelism* are essential aspects of goods flows, it seems sensible to try how Petri nets would perform as a basic concept for a specification tool (compare for instance Magott [16] and Garg [7]). In a recent paper Thomasma and Hilbrecht [23] came to the conclusion that Petri nets provide the most useful approach for specifying material handling control algorithms in flexible manufacturing systems. Di Mascolo et. al. [18] show that Petri nets provide the right language to formulate all sorts of kanban systems in a unified way. So, also the goal of integrating the control process into the specification of physical flows and information flows seems to be best attainable by choosing Petri nets. For a recent overview on properties, analysis and applications of Petri nets, the reader is referred to Murata [20]. For the description of a specification tool for distributed information systems based on Petri nets, the reader is referred to Van Hee, Somers, Voorhoeve [10]. The specification language described in the latter paper indeed leads to executable prototypes. Therefore, the project at the Technical University Eindhoven chose this language, which goes by the name of ExSpect, as basis for the specification tool for goods flow systems. An extensive report on the use of ExSpect for logistic systems together with some extensions for that

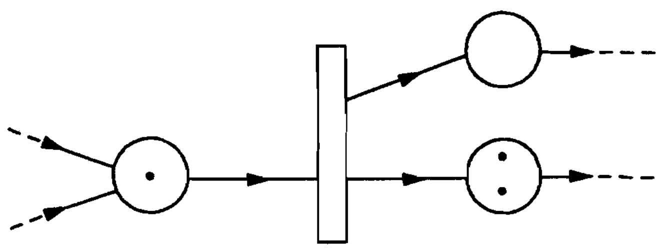

Figure 1: *Part of a Petri net with one token in the input place of the depicted transition and two tokens in one of its two output places.*

purpose will be published as Van der Aalst [2]. A preliminary report is Van der Aalst, Waltmans [4]. One of the proposed extensions is treated in Van der Aalst [1].

In the sequel of this section the basic notations of Petri nets are introduced. This introduction will be informal. We will start with a graphical description, since one of the nice features is that Petri nets allow for a graphical representation:

The basic notations are

transitions,	denoted by	▯
places,	denoted by	○
tokens,	denoted by	•
connections,	denoted by	⟶

A *connection* always connects one *place* with one *transition*. If a connection leads from place *i* to transition *j*, then place *i* is called an *input-place* for transitions *j*. If a connection leads from transition *j* to place *i*, then place *i* is called an *output-place* for transition *j* (see figure 1). A place may contain a number of tokens.

Each connection is supposed to have a *weight*, which is a positive integer. The weights can be inscribed alongside the connection (see figure 2), however weights of size 1 are usually deleted. Any network constructed along these lines is called a Petri net. Petri nets have been introduced to describe dynamic phenomena. So it is important to have rules for the dynamics of Petri nets. In fact, the only dynamic aspect is in the tokens: the distribution of the tokens over the places can change. So the distribution of the tokens over places can be viewed as the state of the Petri net. The rules for a change of the state are the following. If each input place of a transition contains a number of tokens at least equal to the weight of its connection with that transition, then the transition is said to be *enabled* for *firing* (see figure 2). When firing, a transition consumes from each of its input places exactly the number of tokens equal to the weight of the connection. Furthermore, it produces for each of its output places exactly the number of tokens required by the weight of the connection between the transition and the output place (see figure 3).

Indeed, the basic Petri net model as introduced so far reflects the elementary steps of the goods flow process: transitions reflect transformation and/or displacement of goods,

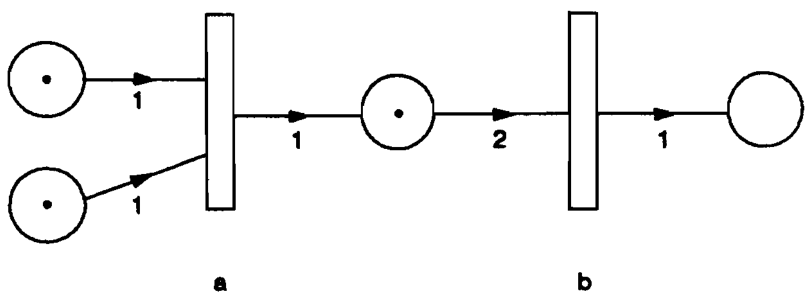

Figure 2: *A Petri net with weights indicated for the connections; transition a is enabled and transition b is not enabled.*

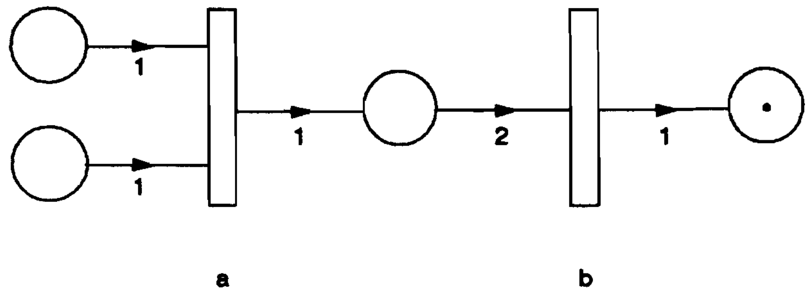

Figure 3: *The Petri net of figure 2 after the firing of transition a and the subsequent firing of transition b.*

whereas places reflect buffering possibilities. In this way the subsequent states of a Petri net may reflect the subsequent positions of a real goods flow system. However, the time durations do not play a role. Therefore, we will introduce time more explicitly in the next section.

3 Specification by timed Petri nets.

For making decisions about goods flow problems, the evaluation of time aspects is crucial in most cases. Time plays a role in essentially two ways. In the first place in the *duration* of transformations and displacement and in the second place in the *availability* of processors for transformation and/or displacement. The second way corresponds in a natural way to the usual way of introducing time in Petri nets (see Murata [20], Zuberek [24]), namely by specifying the time at which a transition might fire. However, in goods flow situations the duration of activities is the most relevant feature, therefore we will introduce time delays due to activities. In fact, also the availability feature can be modelled in this way, so we do not loose modelling power. In the simplest version, each transition gets an attribute *delay* which is a nonnegative real number. In order to process these delays in the right way, all tokens get a *time stamp*, indicating the moment they become available for consumption. It is supposed that a transition becomes enabled as soon as the right numbers of tokens are available in its input places, however, it can only fire at the time indicated by the maximum of the time stamps of the tokens to be consumed. If there are more tokens than necessary, then the ones with the lowest times are consumed.

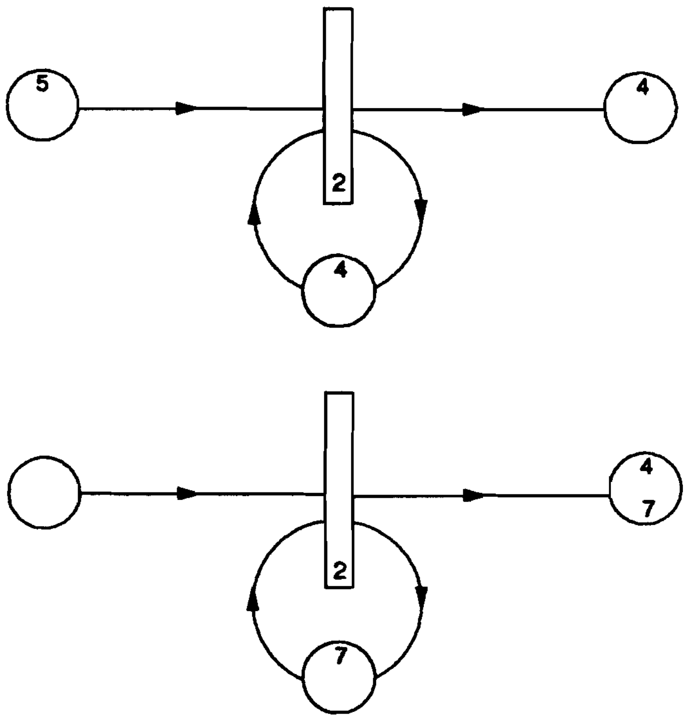

Figure 4: *The transition in the upper timed Petri net is enabled to fire at time 5: the transitions will give a delay of 2 time units. The lower timed Petri net depicts the situation after the firing.*

Note that enabledness of a transition does not necessarily lead to firing of this transition: it is quite well possible that another transition consumes some of its tokens, leading to disabling the first mentioned transition. This way of treating time in Petri nets has been introduced by Van Hee et. al. in [10]. For an illustration of the time procedure see figure 4, where tokens are represented by their time stamp rather than by a dot.

The timed Petri net of figure 4 represents some operation which takes 2 time units and can only perform one execution at a time.

In figure 5 the situation is depicted where the operation is a displacement and the transport vehicle needs some time to return. In figure 6 the same transport operation is depicted now taking into account the loading and unloading operations which require the presence of the vehicle, the load and the fork-lift.

In a similar way a more complex production situation may be described. Figure 7 gives an example of an assembly situation with six types of components coming from the outside. Via three types of subassemblies the final product is reached, which is delivered to the outside world.

In the examples so far the control was of the push type. That means: as soon as jobs and tools are available, the operation is performed. The time stamps of incoming transport jobs (figures 5 and 6) or of incoming components can be exploited to implement the type of control as used in MRP-environments: the timestamps of incoming tokens can then be interpreted as release dates. However, also within the described system there can be some sort of control other than just "push" or "produce if you can." In fact, the specification

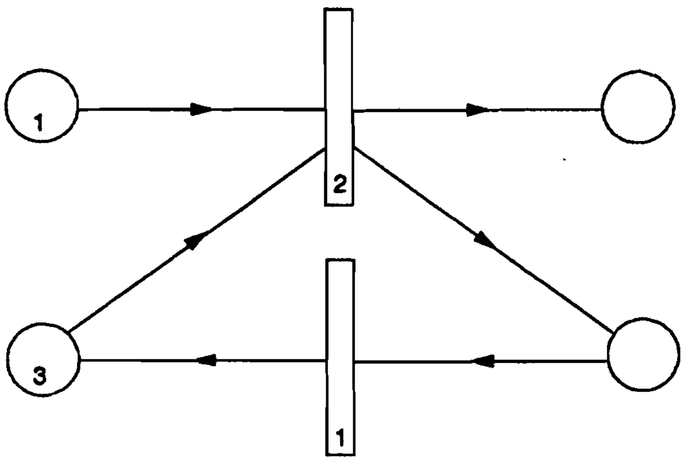

Figure 5: *An operation which requires some dead time after an execution before it can start the next execution, e.g. a transport operation requiring the return of the vehicle involved.*

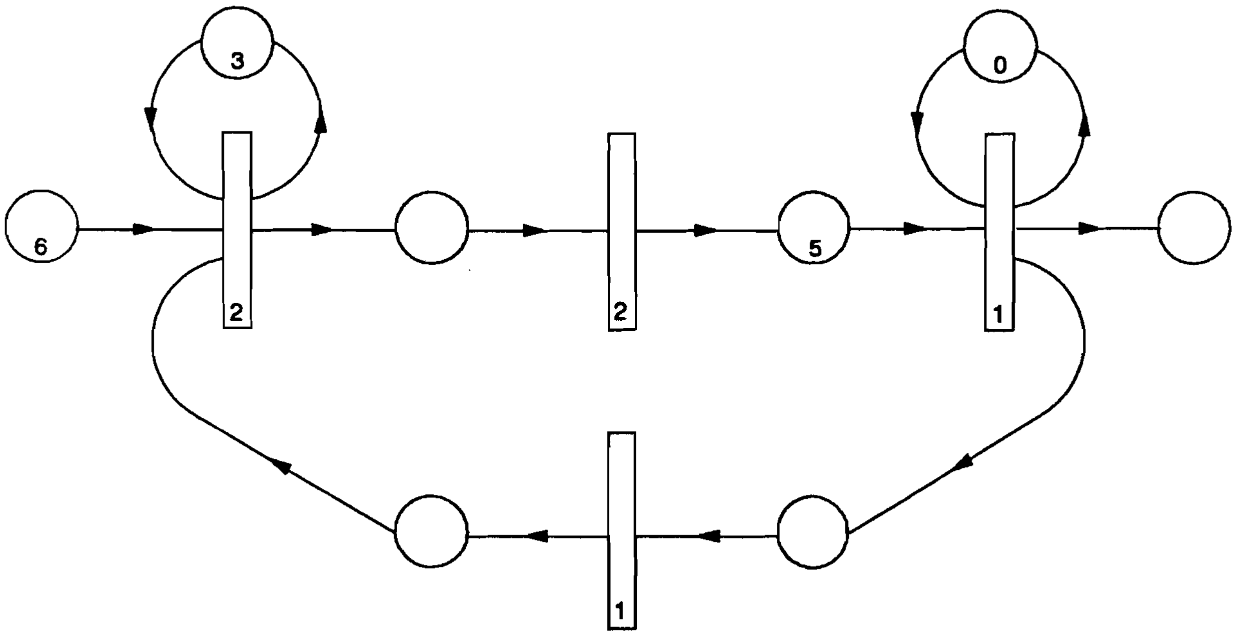

Figure 6: *A transport operation with loading and unloading. The unloading is enabled and may fire at time 5, which will make it possible to start the next loading at time 7.*

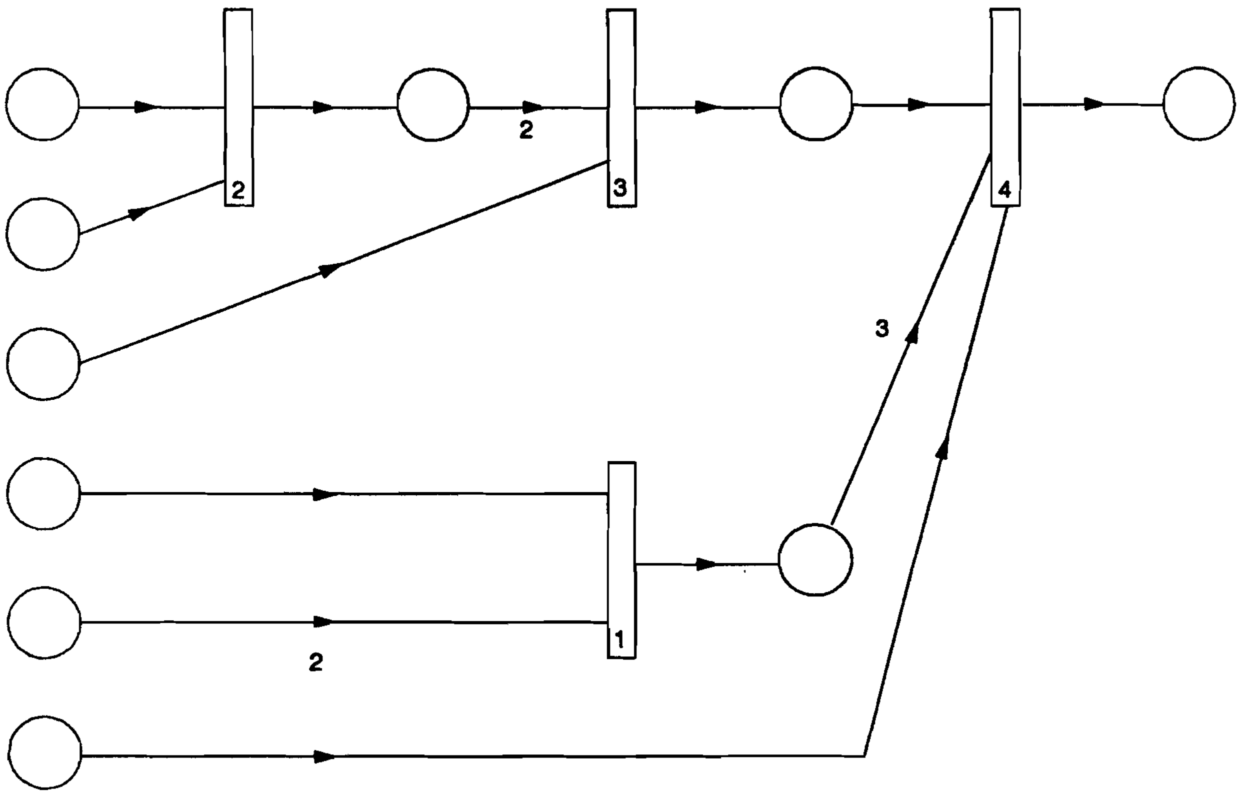

Figure 7: *An assembly situation with six incoming types of components and three types of subassemblies. In some stages there are more specimen of one component or subassembly needed for the next step: this is represented by the weight of a connection; weights of size 1 are not depicted.*

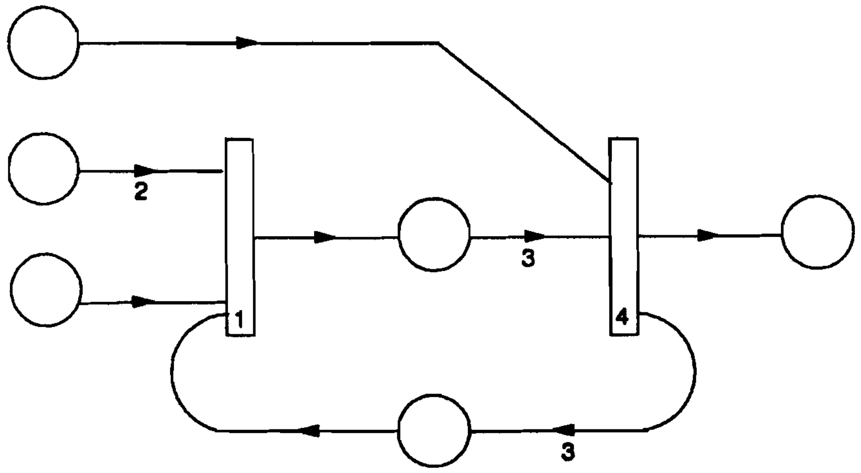

Figure 8: *An assembly system where the final assembly step releases orders (kanbans) for the preceding subassembly. The number of tokens in the cycle is an important design quantity.*

concept as described above is a very natural tool for specifying pull control in the form of kanbans. Figure 8 describes a situation of two subsequent production steps, where the second step needs three units from the first step together with one unit of an external component. The first step is triggered by the second step: each consumption of three units by the second step triggers an order release of three units for the first step. This order can only be executed if the required components are available. In this way the amount of intermediate stock is regulated. Actually, the number of tokens in the cycle determines the performance: with less tokens the second step has more risk of being unable to produce, with more tokens the amount of work in process will be higher (for an extensive treatment of specifying kanban systems by Petri nets, see Di Mascolo et. al. [18]).

For other types of control it would be nice to have another feature in the Petri net concept, namely the feature of having *values* attached to the tokens. In the Petri net literature this feature is usually referred to as *colouring* of the tokens (cf. Jensen [15], Murata [20]). In fact, the introduction of colouring is not absolutely necessary, however, the price of not allowing it would consist in getting very large and relatively unnatural specifications in many situations. In this short overview, we will not treat colouring, but rather refer to Van Hee et. al. [10] and to Van der Aalst [2] for an integrated treatment of the time and colour concepts. In those references a model is introduced which makes it possible that a transition produces tokens with values determined by a function of the values of the consumed tokens. Also the delay of a transition may be a function of the values of the consumed tokens. These features extend the expressive power of the Petri net concept considerably, particularly since the user may define the type of the value rather freely. In this way, it becomes indeed possible to make specifications, which integrate the description of the material flow with the description of the control and of the information flow.

There is still one feature missing so far and that is the feature of uncertainty. Given the way that time has been introduced in Petri nets in this paper, the natural way to introduce uncertainty is to accept probability distributions for the time delay instead of the deterministic values as used so far. Indeed, uncertainty has been introduced in this way in ExSpect, see Van Hee et. al. [10]. For other, strongly related, ways of handling uncertainty, see Murata [20] and Hatono et. al. [9]. With this type of handling uncertainty one obtains Petri net models which are very well suited for simulation. However, as will be explained in Section 5, other ways of evaluation are only feasible for stochastic Petri nets under rather strict conditions. Therefore, it has appeared to be useful to introduce the possibility of only specifying an upper and a lower bound for each delay (see Van der Aalst [1], [2]). Indeed, such a specification is not sufficient for simulation, but it appeared quite well possible to develop other evaluation techniques for Petri nets with *interval time delays* (see Section 5).

4 Hierarchy and modularisation.

The approach as introduced in Section 3 provided an expressive (partly graphical, partly functional) method for specifying goods flow processes including the information flows and the control processes. However, specifications for real systems have a tendency to become rather large and complicated. This is partly due to the preciseness with which the mechanisms can be specified. The main source, of course, is the inherent complexity of modern goods flow processes due to the high performance requirements. Therefore, thinking on goods flow processes always takes place in a more structured way. A specification tool for goods flow processes should support this structured way of thinking rather than hamper it. Support of a structured way of thinking requires the possibility to execute the specification process, or parts of it, in a top-down fashion. Tamura and Hatono [22] describe a hierarchical approach for specifying flexible manufacturing systems by stochastic Petri net models based on a hierarchical structure emerging from the application area.

In the case of the assembly production in figure 7, one can imagine that the two subassembly steps of the top level are performed in one department and the final assembly, together with the remaining subassembly step, is performed in another department. In that case the first two stages in a top-down specification process would result in the representations of figures 9 and 10, where parts of figure 7 are replaced by boxes or rectangles. In our case, where we started with figure 7, the figure 10 may be seen as an aggregation of figure 9 and figure 9 as an aggregation of figure 7. In a top-down specification process, figures 9 and 10 may be seen as stages in this process with the boxes as provisionally unspecified parts.

The feature of specifying top-down in different stages is supported by the tool ExSpect and also the possibility of exhibiting a specification with diminished level of detail. The way of using these properties in specifying goods flow systems is particularly addressed in Van der Aalst [2].

The possibility to distinguish subsystems and treat them separately does help a lot in coping with the complexity and size of a specification. However, it does not diminish the amount of work which is required by a detailed specification. Therefore, it would be necessary to have standard modules available which can be plugged in. For a specific area of application, like goods flow systems, it seems quite well possible to develop standard modules for several activities. When doing so, it is important to choose the right sort of parametrisation. For keeping flexibility it is important to have primarily modules for

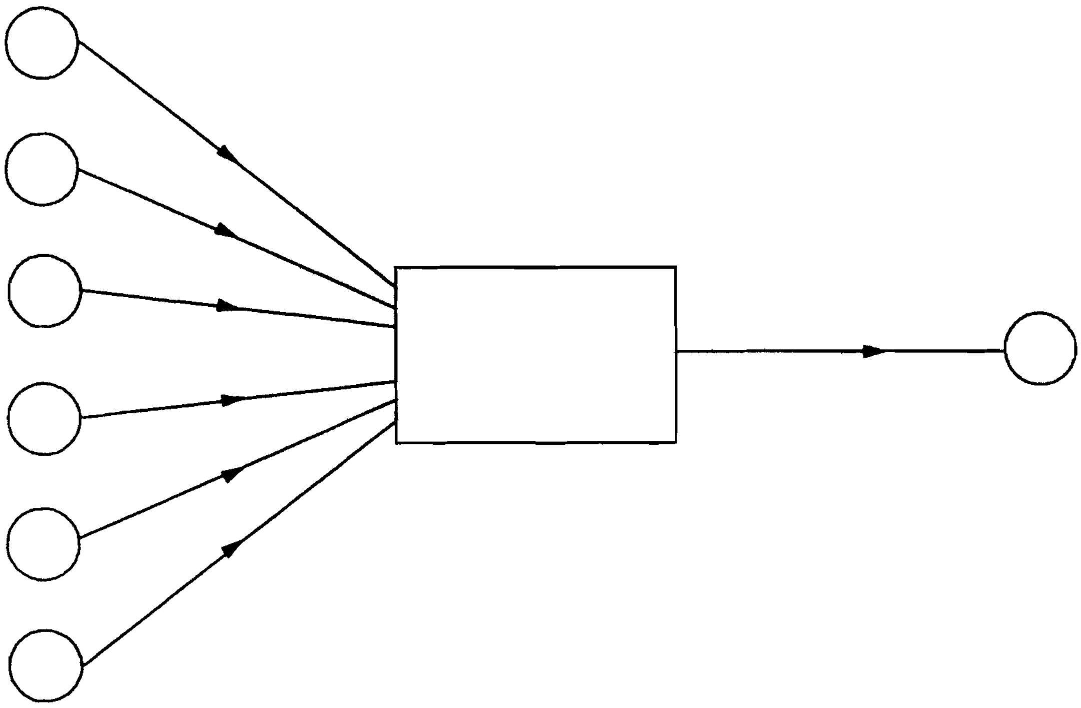

Figure 9: *The assembly situation of figure 7 with the production itself represented by a "black" box.*

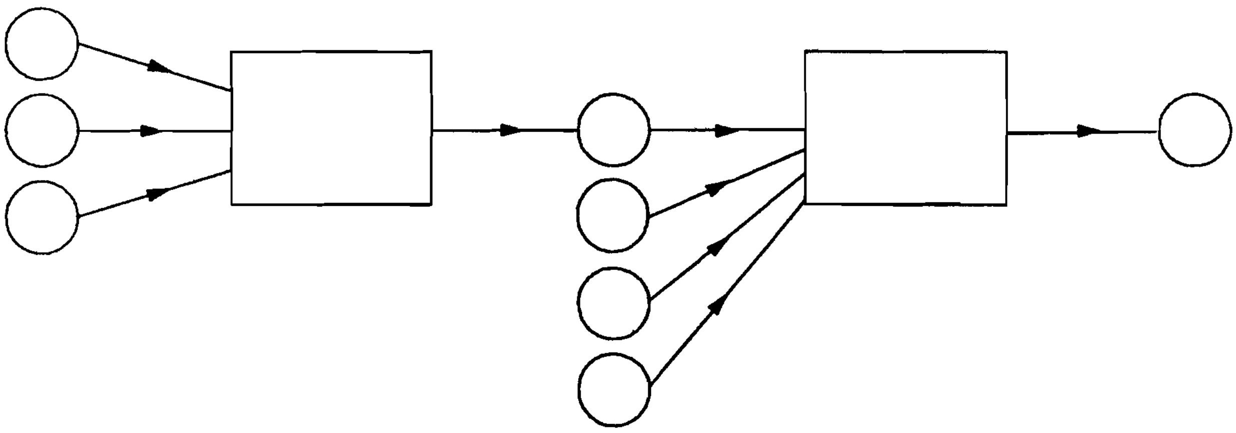

Figure 10: *The assembly situation of figure 7 with boxes representing departments which perform two assembly steps each.*

the elementary activities with regard to handling of goods, information and control. For quick and easy use, it is better to have modules representing more complex activities. In fact, these approaches can be combined by providing in the first place modules for the elementary activities and using these modules to formulate modules for more complex activities. Now the user can choose whether he prefers the flexibility or the easiness of modeling. Eventually, the user can define his own modules for more complex activities. For an elaborate treatment of this topic, the reader is referred to Van der Aalst [2].

5 From specification to analysis.

The material in Sections 2-4 was presented to show that indeed it is quite well possible to make a unified specification method for a large class of goods flow situations while integrating the flows of material and information with the control processes. However, the specification was not a goal in itself. We embarked on this tour, since we concluded in Section 1 that a specification method would be an indispensable tool in a decision support environment. What remains, however, is to show that such a specification method can be used for decision support. In down-to-earth terms this would mean that it should, in the first place, be possible to formulate specifications for alternative scenarios and to compare their performances. In the second place it should be possible to use the specification in a search process for alternatives, i.e. it should be possible to get suggestions for improvements. In Section 1 we concluded that it would be good to decouple specification and analysis. The specification method presented in Sections 2-4 is a result of such a decoupling. The main advantages are

1. the specification method is closely related to the view of the user on the real world;
2. the specification method does not narrow down the class of possible methods of analysis.

However, the price of the last advantage is that the step from specification to analysis becomes a real step, at least if one wants to use standard methods from the O.R.-literature.

In the present section we will deal with the possibilities of using the specifications as developed in Sections 2-4 for evaluating scenarios and possibly suggesting new ones. It should be emphasized that this treatment will be far from complete, primarily since there is still a lot unclear about this topic. So the goal of this section will rather be to indicate possibilities and difficulties than to present full-grown solutions.

Before starting with the exploration it seems sensible to be a little bit more precise on the results of the specification process. Using the tool ExSpect, the user specifies his processes in the form of the figures in this paper on an graphical screen. Via the windowing system the non-graphical information can be added in the form of attributes, tables and functions. The user interface may be primarily graphical, but the resulting specification is represented in the language ExSpect. The user might specify directly in the language, but that is not advisable.

The representation in ExSpect specifies a complete model of reality. Indeed, in ExSpect it is possible to use this representation directly for a simulation. The tool is such that it allows for the addition of a measuring system and the execution of a simulation. Such a simulation may take real-time form, including stops with interventions by the user and restarts. So, performing a simulation is the first and most natural way of evaluating the specified model. For an extensive treatment of this feature, see Van Hee et. al. [10] and Van der Aalst [2]. This is indeed a very nice feature even though the simulation is relatively time consuming.

There are essentially two-different approaches towards analysing the specified situations. The one described in the introduction suggests a library of standard algorithms and tools for making new ones. In that setting one would have to translate the model from the specification language into the form required for algorithmic treatment. In the other approach one tries to develop a library of algorithms which directly treat the specification and do not require a translation step. Until now, the first approach did not get much attention for Petri net type specifications. Therefore, it is not clear how feasible the approach would be. It is clear, however, that some types of translation entail more complications than others. It will definitely be a more straightforward task to translate a Petri net specification to a model for a standard simulation package, than to translate it to a model for a queueing network analyser. However, translation to a model for a queueing network analyser is definitely a more straightforward task than translation to a linear programming model. The closer the structure of the analytic model is to the structure of the specification, the more straightforward the translation and the more likely it becomes that translation can be done (semi-)automatically. It is hoped that these translation processes can be supported by a combination of rule based procedures and user actions.

The second approach, namely developing methods of analysis which use the specification as a an input, got much more attention in the literature. On one hand there are some methods specifically designed for the evaluation of Petri net models and on the other hand some other methods are tailored for treatment of Petri net specifications. In the sequel of this section we will sketch some of the efforts along both lines.

Withing Petri net theory there is, apart from simulation, a lot of attention for the computation of place invariants, transition invariants and reachability graphs (cf. Murata [20]). For decision support with regard to goods flow problems, these notions have only a limited value. Moreover, for practical goods flow problems the required computational effort has a tendency to grow beyond reasonable bounds. Therefore, we will not treat these topics further.

With simulation it is possible to evaluate important aspects of the performance of a design. Simulation is appropriate for evaluating the treatment of a package of orders as well as for evaluating long term behaviour under the influence of a demand generator. In the first type of case statistical aspects are treated by repeated simulations, like in physical experiments. In the second type of case, one long experiment or simulation run may be used to evaluate the statistical aspects of the performance. For these topics see any book on simulation, e.g. Bratley et. al. [5]. However, simulation has some disadvantages. In the first place it is time consuming and in the second place it only evaluates one scenario, without giving much help in finding better scenarios. However, in the last few years results have been published under the heading ***perturbation analysis***, which might be described as a method of estimating the gradient of some performance measure for a discrete event system with respect to a parameter θ, based on a simulation run for one value of θ only. For an introduction to the method and some of its technicalities, see Ho [13], [12] and Heidelberger et. al. [11]. Perturbation analysis might provide a basis for developing techniques for helping in the search process for new and better scenarios or designs. This development is still in its infancy.

The other disadvantage of simulation is that it is rather time consuming. Therefore it would be nice to have analytic methods for doing the same types of performance analysis. For timed Petri nets with deterministic times, there has been quite some research activity in determining cycle times (cf. Murata [20]). Also the analysis of time properties via reachability graphs got some attention (cf. Zuberek [24]). However, for goods flow problems it is important to work with uncertain time delays. If all time delays are negative exponentially distributed, then the dynamic behaviour of the Petri net is described by a Markov process and the analysis of the Markov process can be used for the performance evaluation. However, the condition on the distributions is rather severe and the Markov processes get very large state spaces which tends to make the analyses practically infeasible. For details of this approach and more precise conditions, see Marsan [17], Molloy [19]. A new approach consists of representing the uncertainty of time delays by upper and lower bounds. This approach has been introduced by Van der Aalst in [1] (see also [2]) in order to have models which are more realistic than purely deterministic timed Petri nets, but which remain open for analysis. Indeed, Van der Aalst proposes some methods of analysis which are quite executable and give useful bounds for the time performance. These ideas have been integrated in the tool ExSpect (see [2]). Another quite interesting development is the analysis of Petri net specifications with queuing network techniques. Di Mascolo et. al. [18] treat a restricted class of goods flow situations, namely kanban systems, in this way. They provide an approximative analysis for the related queueing network. As a whole, the developments in this area seem promising.

6 Comments and conclusions.

In the introduction we stressed the importance of having a specification method as basis for a decision support environment. In the Sections 2-4 it was shown how such a specification method for goods flow problems could look like. It appeared that it is quite well possible to develop such a specification method for a large class of goods glow situations and a large class of questions within each situation. Nice features of the method appeared to be in the first place the integration of physical flows, information flows and control processes in one specification and in the second place its completeness which made it possible to use the specification as an executable prototype with which one can execute experiments (i.e. simulations). Perhaps even more important are the expressive power and the close relation of specifications to reality. However, the price to be paid for the advantages consists on one hand of the large size and complexity of a specification and on the other hand, of the difficulty in relating the specifications with algorithmic models of the usual type. The first difficulty is contested by introducing a hierarchical approach and parametrised standard modules. The second difficulty is obviously the most essential one. Until now most effort in this respect has been devoted to attempts to avoid the usual algorithmic approaches and to develop analytical methods specifically tailored for this type of specifications. In doing so, the more usual algorithmic methods have served as a source of inspiration.

However, particularly at this aspect a lot of research has still to be done. In the view of the author, one may expect that rule based methods combined with user interaction will provide the means for translating specifications to typical algorithmic models.

References:

[1] W.M.P. van der Aalst. Interval timed Petri nets and their analysis. *Computing Science Note* 91/09. Technical University Eindhoven, May 1991.

[2] W.M.P. van der Aalst. *Timed coloured Petri nets and their application to logistic systems.* Ph.D. thesis at the Technical University Eindhoven 1992 (to appear).

[3] W.M.P. van der Aalst and A.W. Waltmans. *Modelling flexible manufacturing systems with EXPECT.* p. 330-338 in: B. Schmidt (ed.), Proceedings of the 1990 European Simulation Multiconference. Simulation Councils 1990.

[4] W.M.P. van der Aalst and A.W. Waltmans. *Modelling logistic systems with EXSPECT.* p. 269-288 in: H.G. Sol, K.M. van Hee (eds.), Dynamic modelling of information systems. North-Holland, Amsterdam 1991.

[5] P. Bratley, B.L. Fox, L.E. Schrage. *A guide to simulation.* Springer-Verlag, New York 1987 (second edition).

[6] J. Carlier, Ph. Chretienne, C. Girault. *Modelling scheduling problems with timed Petri nets.* p.62-82 in: G. Rosenberg (ed.), Advances in Petri nets 1984. Springer (LNCS 188), New York 1984.

[7] K. Garg. An approach to performance specification of communication protocols using timed Petri nets. *IEEE Trans. Software Engin.* SE-11 (1985) 1216-1225.

[8] G. Harhalakis, S. Laftit, J.-M. Proth. *Event graphs for modeling and evaluating modern production systems.* p. 438-451 in: G. Fandel, G. Zäpfel (eds.), Modern production concepts; theory and applications. Springer-Verlag, Berlin 1991.

[9] I. Hatono, K. Yamagata and H. Tamura. Modeling and on-line scheduling of flexible manufacturing systems using stochastic Petri nets. *IEEE Trans. Software Engin.* SE-17 (1991) 126-132.

[10] K.M. van Hee, L.J. Somers and M. Voorhoeve. *Executable specifications for distributed information systems.* p. 139-156 in: E.D. Falkenberg, P. Lindgreen (eds.), Information system concepts an in-depth analysis. North-Holland, Amsterdam 1989.

[11] Ph. Heidelberger, X.-R. Cao, M.A. Zazanis, and R. Suri. Convergence properties of infinitesimal perturbation analysis estimates. *Management Science* **34** (1988) 1281-1302.

[12] Y.C. Ho. Performance evaluation and perturbation analysis of discrete event dynamic systems. *IEEE Trans. Automatic Control* AC-32 (1987) 563-572.

[13] Y.C. Ho. Perturbation analysis explained. *IEEE Trans. on Automatic Control* AC-33 (1988) 761-763.

[14] M.A. Holiday and M.K. Venon. A generalized timed Petri net model for performance analysis. *IEEE Trans. Software Engin.* SE-13 (1987) 1297-1310.

[15] K. Jensen. *Coloured Petri nets* in: W. Brauer, W. Reisig, G. Rosenberg (eds.), Advances in Petri nets 1986, Part I: Petri nets, central models and their properties. Springer (LNCS 254), New York 1987.

[16] J. Magott. Performance evaluation of concurrent systems using Petri nets. *Information Processing Letters* **18** (1984) 7-13.

[17] M.A. Marsan. *Stochastic Petri nets: an elementary introduction.* p. 1-29 in: G. Goos, J. Hartmanis (eds.) Advances in Petri nets 1989. Springer-Verlag, New York 1990 (LNCS 424).

[18] M. di Mascolo, Y. Frein, Y. Dallery and R. David. A unified modeling of kanban systems using Petri nets. *Intern. J. Flexible Manufacturing Systems* **3** (1991) 275-307.

[19] M.K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Trans. Comp.* C-31 (1982) 913-917.

[20] T. Murata. Petri nets: properties, analysis and applications. *Proceedings of the IEEE* **77** (1989) 541-580.

[21] M. Silva, R. Valette. *Petri nets and flexible manufacturing.* p. 274-417 in: G. Goos, J. Hartmanis (eds.) Advances in Petri nets 1989. Springer Verlag, New York 1990 (LNCS 424).

[22] H. Tamura and I. Hatono. *Modeling and scheduling of flexible manufacturing systems using timed/stochastic Petri nets.* p. 96-101 in: Proceedings of IFAC Workshop on Discrete Event System Theory and Applications in Manufacturing and Social Phenomena, Shenyang, People's Republic of China, June 25-27, 1991.

[23] T. Thomasma and K. Hilbrecht. Specification methods for material handling control algorithms in flexible manufacturing systems. *Intern. J. Flexible Manufacturing Systems* **3** (1991) 231-250.

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computing Science
PROBABILITY THEORY, STATISTICS, OPERATIONS RESEARCH
AND SYSTEMS THEORY
P.O. Box 513
5600 MB Eindhoven, The Netherlands

Secretariate: Dommelbuilding 0.03
Telephone : 040-473130

-List of COSOR-memoranda - 1991

Number	Month	Author	Title
91-01	January	M.W.I. van Kraaij W.Z. Venema J. Wessels	The construction of a strategy for manpower planning problems.
91-02	January	M.W.I. van Kraaij W.Z. Venema J. Wessels	Support for problem formulation and evaluation in manpower planning problems.
91-03	January	M.W.P. Savelsbergh	The vehicle routing problem with time windows: minimizing route duration.
91-04	January	M.W.I. van Kraaij	Some considerations concerning the problem interpreter of the new manpower planning system formasy.
91-05	February	G.L. Nemhauser M.W.P. Savelsbergh	A cutting plane algorithm for the single machine scheduling problem with release times.
91-06	March	R.J.G. Wilms	Properties of Fourier-Stieltjes sequences of distribution with support in [0,1).
91-07	March	F. Coolen R. Dekker A. Smit	Analysis of a two-phase inspection model with competing risks.
91-08	April	P.J. Zwietering E.H.L. Aarts J. Wessels	The Design and Complexity of Exact Multi-Layered Perceptrons.
91-09	May	P.J. Zwietering E.H.L. Aarts J. Wessels	The Classification Capabilities of Exact Two-Layered Peceptrons.
91-10	May	P.J. Zwietering E.H.L. Aarts J. Wessels	Sorting With A Neural Net.
91-11	May	F. Coolen	On some misconceptions about subjective probability and Bayesian inference.

COSOR-MEMORANDA (2)

91-12	May	P. van der Laan	Two-stage selection procedures with attention to screening.
91-13	May	I.J.B.F. Adan G.J. van Houtum J. Wessels W.H.M. Zijm	A compensation procedure for multiprogramming queues.
91-14	June	J. Korst E. Aarts J.K. Lenstra J. Wessels	Periodic assignment and graph colouring.
91-15	July	P.J. Zwietering M.J.A.L. van Kraaij E.H.L. Aarts J. Wessels	Neural Networks and Production Planning.
91-16	July	P. Deheuvels J.H.J. Einmahl	Approximations and Two-Sample Tests Based on P - P and Q - Q Plots of the Kaplan-Meier Estimators of Lifetime Distributions.
91-17	August	M.W.P. Savelsbergh G.C. Sigismondi G.L. Nemhauser	Functional description of MINTO, a Mixed INTeger Optimizer.
91-18	August	M.W.P. Savelsbergh G.C. Sigismondi G.L. Nemhauser	MINTO, a Mixed INTeger Optimizer.
91-19	August	P. van der Laan	The efficiency of subset selection of an almost best treatment.
91-20	September	P. van der Laan	Subset selection for an -best population: efficiency results.
91-21	September	E. Levner A.S. Nemirovsky	A network flow algorithm for just-in-time project scheduling.
91-22	September	R.J.M. Vaessens E.H.L. Aarts J.H. van Lint	Genetic Algorithms in Coding Theory - A Table for $A_3(n,d)$.
91-23	September	P. van der Laan	Distribution theory for selection from logistic populations.

COSOR-MEMORANDA (3)

91-24	October	I.J.B.F. Adan J. Wessels W.H.M. Zijm	Matrix-geometric analysis of the shortest queue problem with threshold jockeying.
91-25	October	I.J.B.F. Adan J. Wessels W.H.M. Zijm	Analysing Multiprogramming Queues by Generating Functions.
91-26	October	E.E.M. van Berkum P.M. Upperman	D-optimal designs for an incomplete quadratic model.
91-27	October	R.P. Gilles P.H.M. Ruys S. Jilin	Quasi-Networks in Social Relational Systems.
91-28	October	I.J.B.F. Adan J. Wessels W.H.M. Zijm	A Compensation Approach for Two-dimensional Markov Processes.
91-29	November	J. Wessels	Tools for the Interfacing Between Dynamical Problems and Models withing Decision Support Systems.
91-30	November	G.L. Nemhauser M.W.P. Savelsbergh G.C. Sigismondi	Constraint Classification for Mixed Integer Programming Formulations.
91-31	November	J.Th.M. Wijnen	Taguchi Methods.
91-32	November	F.P.A. Coolen	The Theory of Imprecise Probabilities: Some Results for Distribution Functions, Densities, Hazard Rates and Hazard Functions.
91-33	December	S. van Hoesel A. Wagelmans	On the P-coverage problem on the real line.
91-34	December	S. van Hoesel A. Wagelmans	On setup cost reduction in the economic lot-sizing model without speculative motives.
91-35	December	S. van Hoesel A. Wagelmans	On the complexity of post-optimality analysis of 0/1 programs.
91-36	December	F.P.A. Coolen	Imprecise Conjugate Prior Densities for the One-Parameter Exponential Family of Distributions.

COSOR-MEMORANDA (4)

91-37	December	J. Wessels	Decision systems; the relation between problem specification and mathematical analysis.