Putting Petri nets to work in Industry

ELSEVIER

Computers in Industry 25 (1994) 45–54

COMPUTERS IN INDUSTRY

Putting high-level Petri nets to work in industry

W.M.P. van der Aalst

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

Received 27 January 1994; accepted 24 May 1994

Abstract

Petri nets exist for over 30 years. Especially in the last decade Petri nets have been put into practice extensively. Thanks to several useful extensions and the availability of computer tools, Petri nets have become a mature tool for modelling and analysing industrial systems. This paper describes an approach based on a high-level Petri net model, i.e. an extended version of the classical Petri net model. This approach has been used to model and analyze a variety of systems in application domains ranging from logistics to office automation.

Keywords: High-level Petri nets; Applications of Petri nets

1. Introduction

The article "Putting Petri nets to work", written by Agerwala, appeared in 1979 [1]. In this article Agerwala argues that: "Today's modeling tools, appropriate for conventional sequential systems, will be inadequate for the complex concurrent systems of the 80's. Petri nets may offer a solution." Since then Petri nets have become a popular tool for describing and studying concurrent systems. Nevertheless, we believe that the classical Petri net model described in [1] will be inadequate for the complex industrial systems of the 90's. Therefore, we propose a Petri net model extended with 'colour', 'time' and 'hierarchy'.

Automated systems encountered in the fields of logistics, manufacturing, communication and administration have become more complex in the last decade. Hardware and software developments allow for automated systems which are more complex. Moreover, today's systems are often distributed and have to satisfy temporal constraints. Classical Petri nets describing these systems tend to be complex and extremely large. To solve this problem, we have extended the classical Petri net model. First of all, tokens are 'coloured', which facilitates the modelling of objects having attributes. Secondly, we added 'time' to be able to model the temporal behaviour of a system. Finally, we provide a 'hierarchy construct' to decompose complex systems. Petri nets extended with these three features are called *high-level Petri nets*.

In our opinion, high-level Petri nets are suitable for the representation and study of the complex industrial systems of the 90's. The high-level Petri net inherits all the advantages of the classical Petri net, such as the graphical and precise nature, the firm mathematical foundation and the abundance of analysis methods. However, the

SSDI 0166-3615(94)00010-N

practical use of high-level Petri nets and related analysis methods highly depend upon the availability of adequate computer tools. Fortunately, some tools, based on high-level Petri nets, have been put on the market. These tools support the modelling and analysis process. Thanks to these tools high-level Petri nets have been put into practice successfully.

This paper provides an introduction to high-level Petri nets, i.e. the concepts, tools and analysis methods. It also reports experiences gained from a large number of industrial applications.

2. High-level Petri nets

In this paper we use high-level Petri nets to model industrial systems. A high-level Petri net is a Petri net extended with 'colour', 'time' and 'hierarchy'. We start with an informal introduction to the classical Petri net, followed by a short description of each of the extensions.

2.1. The classical Petri net model

Historically speaking, Petri nets originate from the early work of Carl Adam Petri [2]. Since then the use and study of Petri nets has increased considerably. For a review of the history of Petri nets and an extensive bibliography the reader is referred to Murata [3].

The classical Petri net is a directed bipartite graph with two node types called *places* and *transitions*. The nodes are connected via directed *arcs*. Connections between two nodes of the same type are not allowed. Places are represented by circles and transitions by rectangles with a marked corner. (In literature transitions are often displayed as bars.) The marked corner is used to distinguish transitions from subnets, see Section 2.4. Places may contain zero or more *tokens*, drawn as black dots. The number of tokens may change during the execution of the net. A place p is called an *input place* of a transition t if there exists a directed arc from p to t; p is called an *output place* of t if there exists a directed arc from t to p.

We will use the net shown in Fig. 1 to illustrate the classical Petri net model. This figure models a machine which processes jobs and has two states (free and busy). There are four places (cin, free, busy and cout) and two transitions (start and finish). In the state shown in Fig. 1 there are four tokens; three in place cin and one in place free. The tokens in place cout represent jobs to be processed by the machine. The token in place free indicates that the machine is free and ready to process a job. If the machine is processing a job, then there are no tokens in free and there is one token in busy. The tokens in place cout represent jobs which have been processed by the machine. Transition start has two input places (cin and free) and one output place (busy). Transition finish has one input place (busy) and two output places (cout and free).

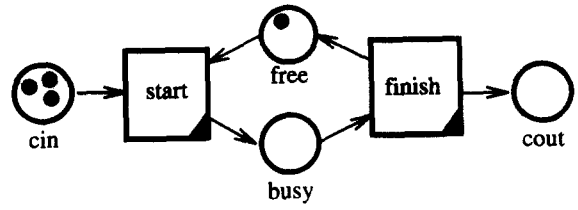


Fig. 1. A classical Petri net representing a machine.

A transition is called *enabled* if each of its input places contains 'enough' tokens. An enabled transition can *fire*. Firing a transition t means consuming tokens from the input places and producing tokens for the output places, i.e. t 'occurs'.

Transition start is enabled in the state shown in Fig. 1, because each of the input places (cin and free) contains a token. Transition finish is not enabled because there are no tokens in place busy. Therefore, transition start is the only transition that can fire. Firing transition start means consuming two tokens, one from cin and one from free, and producing one token for busy. The resulting state is shown in Fig. 2. In this state only transition finish is enabled. Hence, transition finish fires and the token in place busy is consumed and two tokens are produced, one for cout and one for free. Now transition start is enabled again, etc. Note that as long as there are jobs waiting to be processed, the two transitions fire

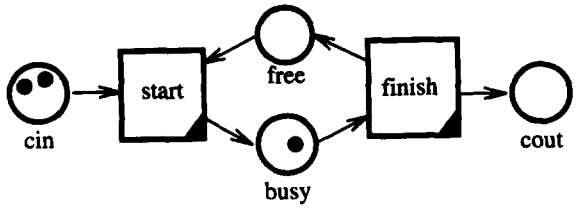


Fig. 2. The state after firing start.

alternately, i.e. the machine modelled by this net can only process one job at a time.

Sometimes there are multiple arcs between a place and a transition indicating that multiple tokens need to be consumed/produced. Consider for example the net shown in Fig. 3. There are two arcs connecting transition rr and input place red; this means that rr is enabled if and only if there are at least two tokens in red. If rr fires, then two tokens are consumed from red and one token is produced for black.

The Petri net shown in Fig. 3 models the following game. The tokens in the places red and black represent red and black balls in an urn respectively. As long as there are at least two balls in the urn, a person takes two balls from the urn. If the balls are of the same colour, then a black ball is returned, otherwise a red ball is returned. Transition rb fires if the person takes two balls having different colours, transition rr fires if two red balls are taken, transition bb fires if two black balls are taken. It can be verified that given an initial state there is precisely one terminal state, i.e. eventually a state is reached where no transitions are enabled.

The classical Petri net model has been used in many application areas, e.g. communication protocols, flexible manufacturing systems and distributed information systems [3]. However, Petri nets describing real systems tend to be complex and extremely large. To solve these problems, many authors propose extensions of the basic Petri net model. We will discuss three of these extensions; 'colour', 'time' and 'hierarchy'. Such extensions are a necessity for the successful application of Petri nets to the modelling of large and complex systems.

2.2. *Adding colour*

Tokens often represent objects (e.g. resources, goods, humans) in the modelled system. Therefore, we often want to represent attributes of these objects. If a truck is modelled by a token in the Petri net, then we may want to represent the capacity, registration number, location, etc. of the truck. Since these attributes are not easily represented by a token in a classical Petri net, we extend the Petri net model with *coloured* or *typed tokens*. In a coloured Petri net each token has a value often referred to as 'colour'. Many coloured Petri net models have been proposed in the literature [4–7]. One of the main reasons for such an extension is the fact that uncoloured nets tend to become too large to handle.

We will use the machine modelled in Fig. 1 to clarify this concept. Tokens in the places cin and cout represent jobs. These jobs may have attributes like an identification number, a description and a due date. We can model this by giving the tokens in cin and cout a *value* (colour) which corresponds to these attributes. In Fig. 4 we see that the job in cin refers to a product TRUCK.A11 and has an identification number 3241 and a due date 29-01-94. The token in place free represents

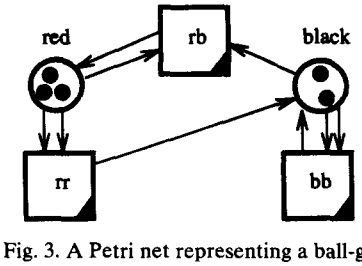


Fig. 3. A Petri net representing a ball-game.

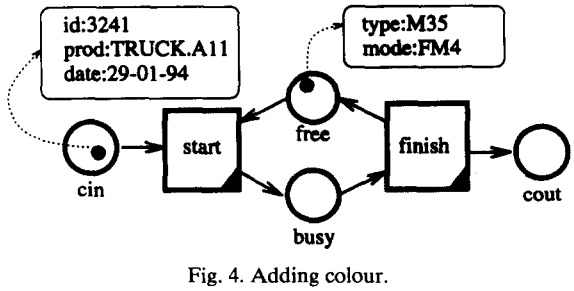


Fig. 4. Adding colour.

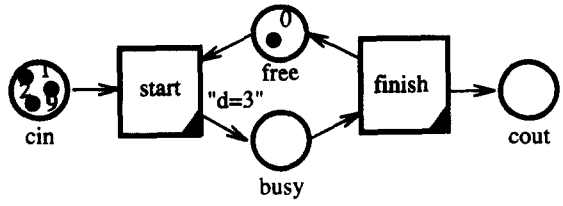


Fig. 5. Adding time.

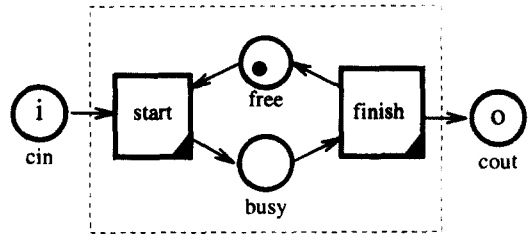


Fig. 6. The definition of the machine system.

a machine and its value contains information about this machine (type and mode).

Transitions determine the values of the produced tokens on the basis of the values of the consumed tokens, i.e. a transition describes the relation between the values of the 'input tokens' and the values of the 'output tokens'. It is also possible to specify 'preconditions', e.g. transition start may have a precondition which specifies that jobs require a machine of a specific type.

2.3. Adding time

For real systems it is often important to describe the *temporal behaviour* of the system, i.e. we need to model durations and delays. Since the classical Petri net is not capable of handling quantitative time, we add a timing concept. There are many ways to introduce time into the classical Petri net [4]. We use a timing mechanism where time is associated with tokens and transitions determine delays.

Consider the net shown in Fig. 5. Each token has a *time stamp* which models the time the token becomes available for consumption. The token in free has time stamp 0, the tokens in cin have time stamps ranging from 1 to 9. Since these time stamps indicate when tokens become available, transition start becomes enabled at time 1. (Time 1 is the earliest moment for which each of the input places contains a token which is available.) Therefore, transition start fires at time 1, thereby producing a token for busy with delay 3. The time stamp of this token is equal to $1 + 3 = 4$. Transition finish will be the next to fire (at time 4), etc. The delay of a produced token can be described by a fixed value, an interval or a probability distribution [3,4,8,9].

2.4. Adding hierarchy

Although timed coloured Petri nets allow for a succinct description of many industrial processes, precise specifications for real systems have a tendency to become large and complex. This is the reason we provide a hierarchy construct, called *system*. A system is an aggregate of places, transitions and (possibly) subsystems.

Fig. 6 shows the definition of the machine system. This system is composed of two places (free and busy) and two transitions (start and finish) and two *connectors* (cin and cout). These connectors provide an interface with the environment of the machine system. The cin connector is an *input connector* (i.e. tokens may enter the system via this connector), cout is an *output connector* (i.e. tokens may leave the system via this connector). If a system is used, the connectors are connected to places at a 'higher level'. Consider for example the net shown in Fig. 7. In this net the same definition is 'installed' three times. In this case, for each of these 'installations' the cin connector is connected to the place arrive

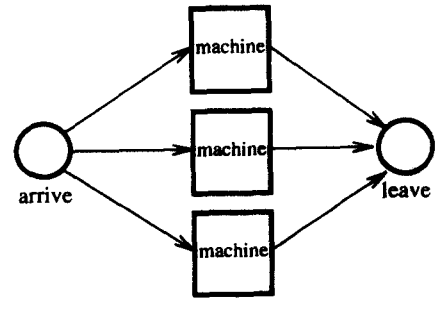


Fig. 7. Three parallel machines modelled in terms of the machine system.

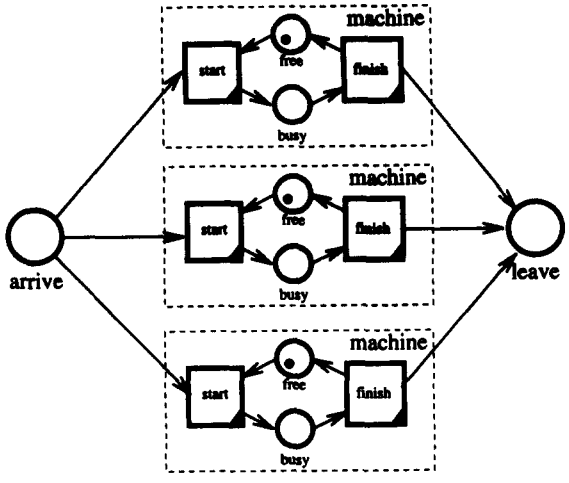


Fig. 8. The 'flat net' corresponding to Fig. 7.

and the cout connector is connected to the place leave, i.e. the connectors of the machine system are 'glued' on top of places at a higher level. If we replace each system by its definition we obtain a 'flat net', i.e. a net without hierarchy. In fact, the semantics of a net with hierarchy are given by the corresponding flat net. Fig. 8 shows the flat net which corresponds to the net shown in Fig. 7.

Both processors and systems are able to consume and produce tokens. In a sense, processors are atomic systems, i.e. systems without an internal state. In other words, a processor is a system without memory. Therefore, we use similar symbols for processors and systems. Both processors and systems are modelled by rectangles and processors have a marked corner to indicate the fact that they are atomic.

The system concept allows for hierarchical modelling, i.e. it is possible to decompose complex systems into smaller subsystems. (Note that it is possible to have an arbitrary number of levels.) For practical applications of Petri nets, the system concept is of the utmost importance. The system concept can be used to structure large specifications. At one level we want to give a simple description of the system (without having to consider all the details). At another level we want to specify a more detailed behaviour.

For a more elaborate discussion on hierarchy constructs, the reader is referred to Jensen [6], Van der Aalst [4,10] and Van Hee [5].

2.5. Language and tools

In the remainder of this paper we will refer to Petri nets extended with 'colour', 'time' and 'hierarchy' as *high-level Petri nets*.

Only a few high-level Petri net models (i.e. hierarchical timed coloured Petri net models) have been proposed in literature. Even fewer high-level Petri net models are supported by software tools. Nevertheless, there are at least two software products, *ExSpect* [10–12] and *Design / CPN* [6], that are being distributed on a commercial basis. Both software products provide a graphical interface to create, modify and simulate high-level Petri nets. Moreover, they provide analysis tools and reporting facilities.

To specify the behaviour of each transition (i.e. the number of tokens produced and the value and delay of each produced token), Design/CPN provides an 'inscription language' (expressions on the input and output arcs of a transition). ExSpect (Executable Specification Tool) uses a 'Z-like' specification language [13] to describe the behaviour of a transition. Both languages originate from 'pure' functional languages.

The approach described in this paper is based on the software package ExSpect. ExSpect has been developed by the Information Systems Department of Eindhoven University of Technology. Since 1992, ExSpect is being distributed by Bakkenist Management Consultants. [1]

2.6. Analysis of Petri nets

The complexity of the design and control problems encountered in modern industrial systems is increasing. Therefore, we need methods and techniques to support both the modelling *and* analysis of these systems. High-level Petri nets

[1] Bakkenist Management Consultants, Wisselwerking 46, 1112 XR Diemen, The Netherlands.

often allow for a representation which is close to the problem situation, i.e. it is possible to model the system in a natural manner. This representation can be used as a starting point for various kinds of analysis. In a sense, the Petri net representation serves as an interface between the problem situation and the method(s) of analysis. In fact, high-level Petri nets provide a 'solver-independent' medium that can be used to make a concise 'blueprint' of the industrial system we want to analyze. This blueprint may be used at different levels of decision making and can be used as a starting point for various means of analysis. Compared to the usual algorithmic approaches (where the emphasis is on the analysis process rather than the modelling process), our approach is characterized by the fact that during the modelling process the user is not shackled by the techniques which are going to be used to analyze the model.

For an overview of the many analysis methods developed for Petri nets the reader is referred to Jensen [7], Murata [3], Silva and Vallette [14] and Van der Aalst [4,15]. These methods can be used to prove properties (safety properties, invariance properties, deadlock, etc.) and to calculate performance measures (response times, waiting times, occupation rates, etc.). In this way it is possible to evaluate alternative designs.

3. Applications

High-level Petri nets have been used in many application areas, e.g. communication protocols, flexible manufacturing systems, computer systems, production systems, logistic systems, administrative systems, real-time systems, work-flow systems, and distributed information systems. In The Netherlands, the Petri net based tool ExSpect has been applied to the modelling and analysis of a variety of systems. This section briefly discusses some of these applications.

3.1. Prototyping of software

High-level Petri nets have been used for the specification of software, ranging from operating systems to decision support systems. Petri nets are able to represent the flow of control in a program in a straightforward manner. Moreover, Petri nets can clearly and explicitly represent the interaction between concurrent processes [1]. By specifying a piece of software in terms of a high-level Petri net, it is possible to analyze the corresponding specification by means of Petri net based analysis techniques. High-level Petri nets are 'executable', i.e. a computer can generate a sequence of states $s_0, s_1, \ldots, s_n$ such that s_0 is the initial state and s_{i+1} is reachable from s_i by firing a transition. Therefore, it is possible to simulate the system modelled by the high-level Petri net. Many Petri net based tools (e.g. ExSpect) allow the user to interact with a running simulation. The user is allowed to add and remove tokens, change values of tokens, etc. Therefore, it is possible to obtain a rudimentary *prototype* by specifying a program in terms of a high-level Petri net. ExSpect also allows for the definition of a customized user interface; this way it is possible to create a tailor-made prototype for a specific application. Such a prototype can be used to validate the specification by future users of the program.

In the Esprit project PROOFS (EP 5342) [16], high-level Petri nets have been used to prototype distributed applications for companies such as Alcatel, Eritel, Prisma and Telesystems. Typical applications prototyped in PROOFS are traffic control systems, banking systems and conferencing systems. One of the main results of this project was the 'PROOFS method' [16], a coherent set of guidelines and techniques to support the development of distributed applications by means of high-level Petri nets.

PROOFS pointed out a major problem; at the moment there is no smooth transition possible from the specification phase to the implementation phase. The process of transforming formal specifications into efficient machine code is done mainly by hand. Therefore, automatic code generation would be desirable. In PROOFS some experience with code generation has been acquired. Since code generation would surpass the need for a separate coding phase (at least to a large extent), future research and developments should focus on this aspect.

3.2. (Re)design of logistics and manufacturing systems

Over the last decade, logistics has become an important issue in many organizations. This is a direct consequence of the fact that modern organizations are required to offer a wide variety of products, in less time and at reduced prices. To support the (re)design of parts of these organizations, there is a need for an integrated framework for the modelling and analysis of logistics and manufacturing systems. The Petri nets described in this paper provide such a framework. High-level Petri nets can be used to represent logistics and manufacturing systems in a very natural manner; goods and capacity resources are represented by tokens, buffers, storage space and media are represented by places, and operations are represented by transitions. Petri nets are well suited to model flows of goods, resources and information in a unifying way. Modelling these flows by tokens seems very natural. A place either represents a medium through which something is sent or some storage space (i.e. a buffer). The fact that flows are represented graphically is a very important quality, since it makes the overall structure comprehensible and supports the communication between people having different backgrounds. Moreover, tools such as ExSpect allow for the animation of systems modelled in

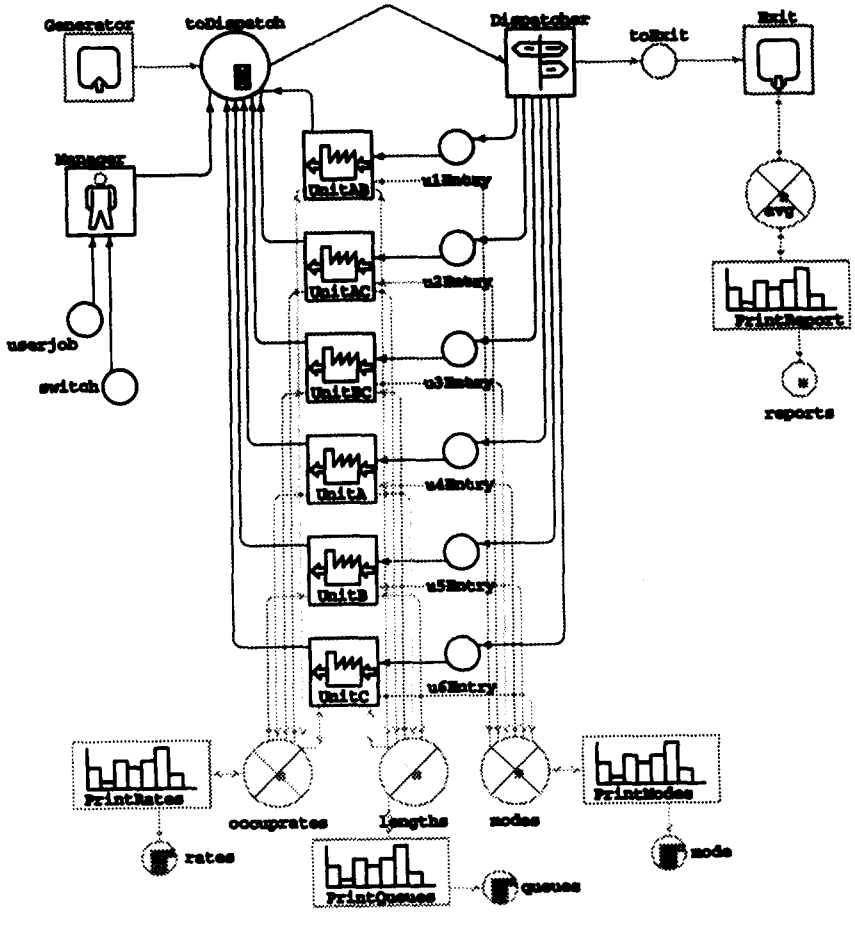


Fig. 9. Animation of a job-shop modelled in terms of a high-level net.

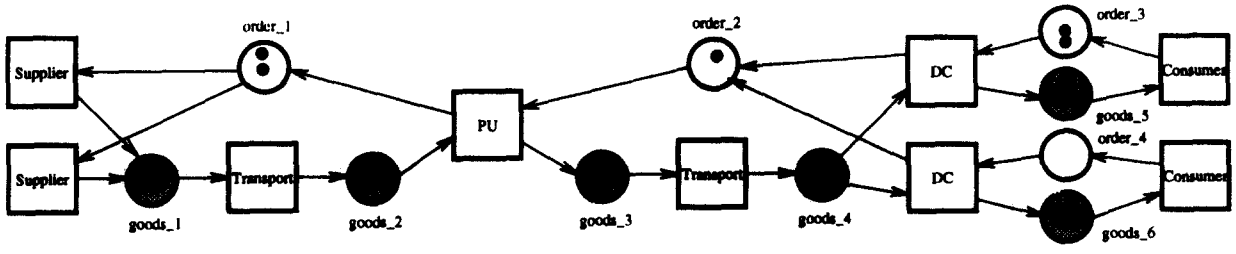


Fig. 10. A logistic chain modelled with logistic building blocks.

terms of a Petri net (see Fig. 9). For a more detailed discussion on this subject the reader is referred to [4,17].

ExSpect has been used to model many logistic systems. The TASTE project [18] resulted in the development of a comprehensive set of logistic building blocks. These building blocks have been used to model distribution processes of Dutch companies such as Ahold, Unilever, DAF, Bührmann-Tetterode, etc. There are building blocks to model transport, supply, demand, production and stock-holding processes. Fig. 10 shows a logistic chain modelled in terms of these building blocks. Note that each building block corresponds to a parameterized system definition. Consider for example the PU system shown in Fig. 10. This building block has two input connectors, two output connectors and parameters to specify the characteristics of a particular production unit (bill-of-material, resources, capacity, suppliers, control strategy, etc.).

3.3. (Re)design of administrative organizations

There are many similarities between logistics and administrative processes. A logistic system manages the flow of goods, an administrative system manages the flow of documents. Both systems aim at a reduction of throughput times and resources. These similarities explain the term 'office logistics'.

Modelling an administrative organization in terms of a Petri net allows for the study of its efficiency, performance and flexibility. This way it is possible to evaluate alternative designs without doing 'real' experiments.

Workflow management systems are receiving more and more attention. It is becoming clear that workflow management systems are the next step in supporting office work, after other tools like database management systems, spreadsheets and electronic mail systems. There are several commercial workflow management systems on the market and experiments have shown that they are successful. High-level Petri nets seem to be a suitable tools for the modelling and analysis of workflow and workflow management systems [19,20].

3.4. Traffic control

The Dutch Railway Company (NS) is currently involved in a number of large development projects. The goal of these projects is to improve the management of trains. To be able to evaluate these costly development projects, NS is looking for tools and techniques to determine the quality of the infrastructure and the traffic control. One of the main tools that has been selected for this purpose is the software package ExSpect. ExSpect has been used for two purposes: (1) the scheduling of train movements between railway stations and (2) the analysis of train management in large railway stations. This way it was possible to analyze and prototype alternatives in a very short time. For NS, ExSpect has become one of the standard tools for the evaluation of large development projects.

4. Conclusion

In this paper we have indicated that high-level Petri nets, i.e. Petri nets extended with 'colour',

'time' and 'hierarchy', can be used for the modelling and analysis of many complex systems encountered in industry. Modelling a system in terms of a high-level Petri net has a number of potential advantages. First, the graphical nature of high-level Petri nets allows for models that are easy to understand. Secondly, high-level Petri nets have formal semantics, thus leading to precise and unambiguous descriptions. Thirdly, the behaviour of the modelled system can be analyzed using Petri net theory. Finally, there are a number of software packages available for the modelling and analysis of systems in terms of high-level Petri nets. Note that each of the extensions is crucial for the successful application of Petri nets in industry. If we omit one of these extensions, it will be difficult to model certain aspects. For example, if we omit 'time', it becomes hard to model the temporal behaviour of many 'real' systems.

High-level Petri nets have been used to model a variety of industrial systems. In this paper we mentioned some of these applications, e.g. prototyping of software, (re)design of logistic systems, (re)design of administrative organizations. These applications point out one of the merits: high-level Petri nets are a uniform design language. The same language is used for the modelling of systems in hardware, software, production, distribution, transportation, and office environments. Moreover, high-level Petri nets can be used at many levels of abstraction. The fact that high-level Petri nets are applicable across the entire spectrum makes it a common description language having great potential.

References

[1] T. Agerwala, "Putting Petri nets to work", *IEEE Comput.*, Vol. 12, No. 12, December 1979, pp. 85–94.

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

[3] T. Murata, "Petri nets: Properties, analysis and applications", *Proc. IEEE*, Vol. 77, No. 4, April 1989, pp. 541–580.

[4] W.M.P. van der Aalst, Timed coloured Petri nets and their application to logistics, PhD thesis, Eindhoven University of Technology, Eindhoven, 1992.

[5] K.M. van Hee, *Information System Engineering: A Formal Approach*, Cambridge University Press, 1994.

[6] K. Jensen, "Coloured Petri nets: A high level language for system design and analysis", in: G. Rozenberg (ed.), *Advances in Petri Nets 1990* (Lecture Notes in Computer Science, Vol. 483), Springer-Verlag, New York, 1990, pp. 342–416.

[7] K. Jensen, *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use* (EATCS Monographs on Theoretical Computer Science), Springer-Verlag, New York, 1992.

[8] M. Ajmone Marsan, G. Balbo and G. Conte, "A class of generalised stochastic Petri nets for the performance evaluation of multiprocessor systems", *ACM Trans. Comput. Syst.*, Vol. 2, No. 2, May 1984, pp. 93–122.

[9] B. Berthomieu and M. Diaz, "Modelling and verification of time dependent systems using time Petri nets", *IEEE Trans. Software Eng.*, Vol. 17, No. 3, March 1991, pp. 259–273.

[10] W.M.P. van der Aalst and A.W. Waltmans, "Modelling logistic systems with ExSpect", in: H.G. Sol and K.M. van Hee (eds.), *Dynamic Modelling of Information Systems*, Elsevier, Amsterdam, 1991, pp. 269–288.

[11] W.M.P. van der Aalst and A.W. Waltmans. "Modelling flexible manufacturing systems with ExSpect", In: B. Schmidt (ed.), *Proc. 1990 Eur. Simulation Multiconf.*, Nürnberg, June 1990. Simulation Councils Inc., 1990, pp. 330–338.

[12] K.M. van Hee, L.J. Somers and M. Voorhoeve, "Executable specifications for distributed information systems", in: E.D. Falkenberg and P. Lindgreen (eds.), *Information System Concepts: An In-depth Analysis* (Proc. IFIP TC8/WG8.1 Working Conf., Namur, Belgium, 1989), Elsevier, Amsterdam, 1989, pp. 139–156.

[13] J.M. Spivey, *The Z Notation: A Reference Manual*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[14] M. Silva and R. Valette, "Petri nets and flexible manufacturing", in: G. Rozenberg (ed.), *Advances in Petri Nets 1989* (Lecture Notes in Computer Science, Vol. 424) Springer-Verlag, New York, 1990, pp. 274–417.

[15] W.M.P. van der Aalst, "Interval timed coloured Petri nets and their analysis", in: M. Ajmone Marsan (ed.), *Application and Theory of Petri Nets 1993* (Lecture Notes in Computer Science, Vol. 691), Springer-Verlag, New York, 1993, pp. 453–472.

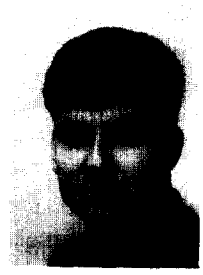
[16] W.M.P. van der Aalst, K.M. van Hee, N. Tréves and R. di Giovanni, PROOFS: formalisms and methods, TUE-TR-0035-V4.0-WP1, Eindhoven University of Technology, Eindhoven, 1993.

[17] W.M.P. van der Aalst, "Modelling and analysis of complex logistic systems", in: H.J. Pels and J.C. Wortmann (eds.), *Integration in Production Management Systems* (IFIP Transactions, Vol. B-7), Elsevier, Amsterdam, 1992, pp. 277–292.

[18] W.M.P. van der Aalst, M. Voorhoeve and A.W. Waltmans, "The TASTE project", in: *Proc. 10th Int. Conf. on Applications and Theory of Petri Nets*, Bonn, June 1989, pp. 371–372.

[19] C.A. Ellis and G.J. Nutt, "Modelling and enactment of workflow systems", in: M. Ajmone Marsan (ed.), *Application and Theory of Petri Nets 1993* (Lecture Notes in Computer Science, Vol. 691), Springer-Verlag, New York, 1993, pp. 1–16.

[20] W.M.P. van der Aalst, K.M. van Hee and G.J. Houben, "Modelling workflow management systems with high-level Petri nets", in: *Proc. 2nd Workshop on Computer-Supported Cooperative Work, Petri Nets and Related Formalisms*, Zaragoza, 1994, pp. 31–50.



Wil van der Aalst is an Assistant Professor in the Department of Mathematics and Computing Science at Eindhoven University of Technology. In 1992, he completed his PhD thesis on Petri nets and their application to logistics. His research interests are in simulation, information systems, manufacturing and real-time systems. He is also working as a part-time consultant for Bakkenist Management Consultants.