Tools for the interfacing between dynamical problems and models within decision support systems

TECHNISCHE UNIVERSITEIT EINDHOVEN
Faculteit Wiskunde en Informatica

Memorandum COSOR 91-29

Tools for the Interfacing Between Dynamical Problems and Models within Decision Support Systems

J. Wessels

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

Eindhoven, November 1991
The Netherlands

Tools for the Interfacing Between Dynamical Problems and Models within Decision Support Systems

Jaap Wessels
International Institute for Applied Systems Analysis
Laxenburg, Austria
Technical University Eindhoven
Eindhoven, The Netherlands

November 4, 1991

Abstract

This paper particularly addresses the difficulties arising from formulating models for dynamical problems in such a way that they can be treated by appropriate solvers. One of these difficulties is that different types of solvers are recommendable for different questions within the same situation. Therefore it is recommended to use solver-independent specification methods. Since specification and respecification can be time-consuming and boring, it is also recommended to develop rule-based specification tools. The paper is illustrated with a sketch of possible specification methods for manpower policy making and for goods flow control.

1 Introduction

There is a big difference in requirements between decision support systems for operational planning on one hand, and for tactical and strategic decision making on the other. Of course it would be nice if one could develop systems which were useful for both purposes, however, it is already difficult to satisfy the requirements for one purpose only. In the present paper the subject will be tactical and strategic decision making, rather than operational planning. For tactical and strategic decision making the decisions usually regard processes which unfold in an uncertain and partly unknown environment. The processes themselves may also feature uncertainty. Depending on the question, one may use a static, deterministic model or, if necessary, introduce dynamics and/or uncertainties into the model. Decision support systems should be such that they facilitate different types of analysis with different types of models based on the same real-life process. Therefore it is essential to have a decoupling of process description, mathematical models and analytic methods with, of course, good tools for easy transition from one to the other.

In the present paper we will give a short introduction into this problem area with a short overview of the approaches which might be helpful (alone or in combination). There will be particular attention for the possibility of using process specification systems in the form of languages or otherwise. The point of view will be illustrated by introducing a specification language which can be used in goods flow problems and a generic model which is useful for process description in manpower policy making.

The set-up of the paper is as follows. In Section 2 the problems as described above will be worked out in more detail. Section 3 is devoted to a short description of ways out for these problems. In Section 4, the generic model for manpower policy making is worked out in some detail and it is argued that rule-based methods may be helpful in overcoming the big load of specification work, but also in designing a solver. In Section 5, a specification language will be introduced for goods flow problems. Finally, Section 6 will contain some comments and conclusions.

2 From Problem to Analysis and Back

The basic loop in attacking decision problems with formal methods consists of three activities and two intermediate stages as depicted in Figure 1. In making a decision analysis for a tactical or strategic problem, this loop is usually passed through several times. In the case of "what if" analyses the number of passings may be quite substantial. Actually, if dynamics are an essential part of the model, then such scenario analyses are more the rule than the exception, since only for very well-structured models it is feasible to analyse a model for which the search for good decisions is integrated in the formal analysis.

For subsequent passings of the basic loop, the model and algorithm used may be essentially the same. Apparently, the simplest situation is that only some parameters in the model are changed and the same algorithm is used. However, particularly for tactical and strategical problems, this is not always the case. For example, if the problem consists of designing a new logistic concept for a manufacturing company or redesigning the production facilities for a department of the company, then it is quite sensible to search by changing parameters for a good setting within one logistic or technical concept. However, at some stage it will be clear what is reachable in costs and performance with the concept at hand. Then it is the right time to compare the results with the results of another concept. Actually, when designing a logistic concept (cf. Bertrand, Wortmann, Wijngaard [5]), it is often necessary to zoom in on a particular link of the chain in order to check in more detail whether this link can deliver the required performance or not. Similarly, for the technical design problem, it may be necessary to check how a technical concept would allow this particular department to function as a link in the chain.

Technically speaking, subsequent passings of the basic loop may show changes with respect to the following features:

- parameter values
- dimensions

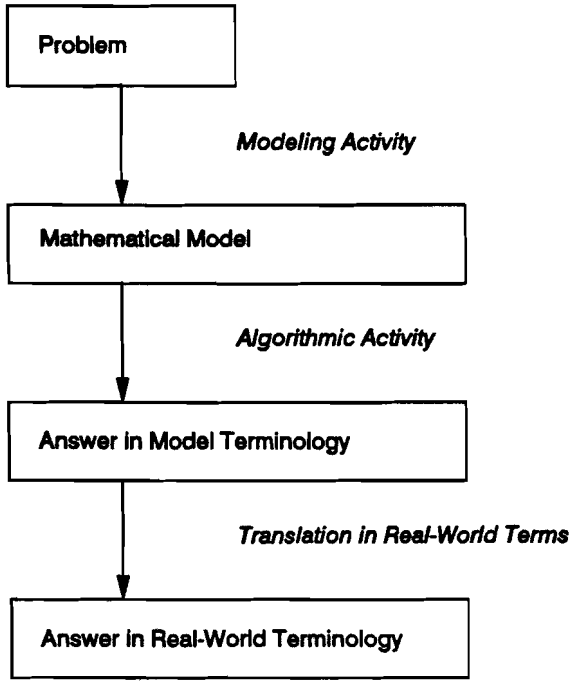

Figure 1: *The basic loop of a decision analysis with mathematical means.*

- the level of detail (aggregation–disaggregation)
- the subject of emphasis (decomposition–composition)
- the method of analysis

It is quite common that in the first stages of a design problem of one of the types mentioned above, a rough model may be evaluated using some type of mathematical technique (e.g. from queueing or inventory theory), whereas in latter stages it may only be feasible to evaluate the more precise model by simulation. A major advantage of applying mathematical techniques of analysis is that they usually provide a direct functional relation between the process parameters and the performance characteristics (one of the nicest examples is the Pollaczek–Khintchine formula for waiting times at $M|G|1$-queues, cf Kleinrock [9], which may be used frequently in design problems for the flow of goods). The major weak point for analytical techniques is that they usually work only for simple well-structured problems and that is exactly why simulation often comes in at some stage of the analysis (note that simulation is one of the most frequently used OR-techniques in practice; compare for instance Tilanus [13]). The more essential model changes (and also the possible change of algorithm) complicate the automatic execution of the three activities in the basis loop considerably, even if one admits that not necessarily every step in the execution should be automated. Below we summarize the major complications for the three activities of the basic loop:

a. The modelling activity. The format of the model depends heavily on the type of algorithm to be used (e.g. when designing a distribution system one may execute an analysis based on capacities using linear programming, but also perform a

day-to-day simulation to find bottlenecks in local time). Even in apparently simple situations, the amount of required data can be considerable. In several cases this already holds for new scenarios which do not differ essentially from the old ones. This is particularly true if small changes generate a whole cascade of other changes which bring about a lot of mainly (but not only) clerical work. It may be difficult to automate this mainly clerical work since it requires at least some problem knowledge (a good example of this difficulty is in decision analysis for manpower policy making).

b. **The algorithmic activity.** An alternative scenario or an alternative level of detail or a change of emphasis may require an alternative way of analysis, since the old one would not work anymore or becomes practically infeasible or just inefficient (in both previously mentioned examples this feature may occur; if we consider the distribution problem, then capacity problems may be attacked by linear programming, but capacity assigned in this way might still cause difficulties and therefore it might be sensible to analyse a possible bottleneck as a queueing problem and decide on some final adjustments using a simulation model for the different interacting parts of the distribution system).

c. **The translation activity.** The translation from model results to real-life conclusions is complicated by the fact that a large assortment of different types of models can play a role. This complication is the natural counterpart of the first complication mentioned for the modelling activity.

3 Practical approaches for the complications.

In this section we will characterize approaches which can be used (and are used in practice) to solve the complications as mentioned in the previous section. The real solution usually consists of a mixture of the approaches which will be presented. The effective mixture heavily depends on the situation.

a. *Choosing a well-defined split-up of decision levels and selecting a flexible type of solver for each level.* This approach belongs to the category divide-and-conquer. The structure defined by the hierarchial split is rather rigid, but for each level it is quite possible that flexibility is attained. A result of this approach can be that it becomes simple to replace the solver for a certain level. In designing structures for the distribution of goods, this approach is quite accepted (see for instance Benders and Van Nunen [4]). A nice example of this approach is found in Zijm [18], where it indeed appears to be possible to make solvers flexible by applying simulated annealing for different levels. Simulated annealing is certainly not the most efficient technique for the relevant discrete optimization problems, but it is clearly very flexible in the sense that it simply adapts if constraints are added or deleted. The more efficient heuristics would require an expert for designing a variant in each new case. For operational decision problems, like the last mentioned example of Zijm, this approach has more chances than for tactical and strategic decision making, since the rigidity of the hierarchical structure is less squeezing for operational problems and also the requested variants of models on one level are likely to be solvable with one solver. Nevertheless, every part of this approach which can be executed is useful.

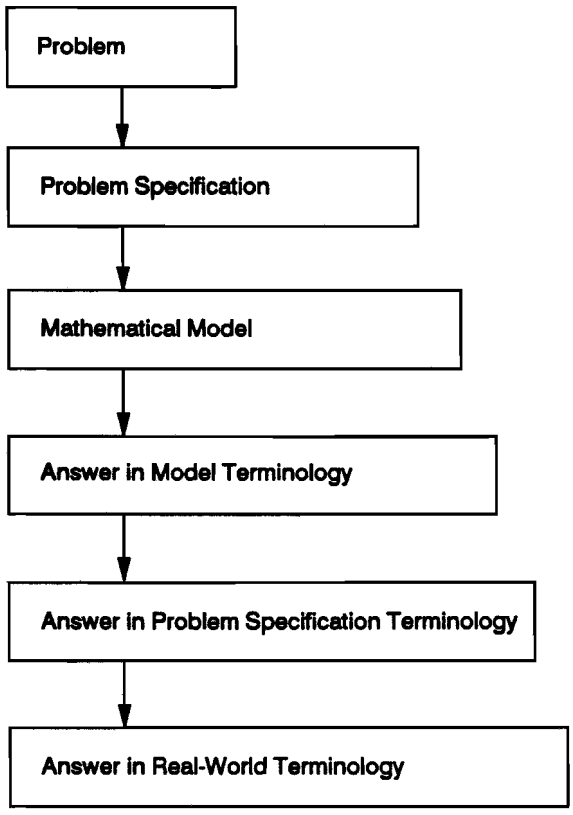

Figure 2: *The stages for the extended basic loop.*

b. *Developing a problem specification method without any reference to methods of analysis.* This might be seen as another way of divide-and-conquer. It decouples the formal description of the problem from the formulation of a model for analysis. A first result is that the formal description of the problem can be executed in a way which is close to the world of the problem owner instead of close to the work of the mathematical analyst. However, the most important result is that such a formal problem description can be used as a basis for formulating different types of models for mathematical analysis. This may regard models for some problem specific method of analysis, but also powerful library methods for standard mathematical models, like linear programming. The more problem-oriented the specification method is, the more easy it becomes for the user or for the user–system interface, however, the more complicated it becomes to automate the translation of problem description to model for analysis. Using this approach the basic loop of Section 2 (Fig. 1) is replaced by the loop of Figure 2, where the first activity (making a formal problem specification) will usually depend heavily on the specification of the previous round and the second activity (making a mathematical model) can hopefully be automated largely or even completely. There is even a tendency to skip the two inner stages of the loop in Figure 2 and develop special methods for analyzing the problem specification directly. In the following section the advantages and disadvantages of the latter approach will be discussed in more detail. Technically speaking, methods for problem specification are always based on a generic model for describing the process mechanisms. Such models may be enhanced by qualitative features for indicating the relative hardness of constraints or the relative importance of goals. The

complete problem specification is therefore not necessarily a mathematical model. However, to be useful it should be very close to it and the qualitative elements should be formalized in such a way that they can be used to make model choices, preferably, in an automated way, but, possibly based on subjective judgement. In the most far-reaching form, a language may be designed to formulate specifications in. In Section 4, an example will be treated based on a generic model, where the specification process is menu driven. Section 5 will be devoted to a language which can be used for goods flow situations.

c. *Using rule-based methods for executing the activities.* The problem of having activities which are more complicated than usual within purely operational decision support systems may in several cases be tackled by introducing rule based methods in the first phase of the execution of an activity. Let us consider the basic loop in the extended form of Figure 2. The last two activities of that loop are counterparts of the first two and therefore can be left out of this discussion. The remaining three activities are:

- making a formal problem specification;
- choosing a model type and specifying the model;
- choosing an algorithm (possibly also making it) and executing it.

It will be clear that all three activities consist of several subactivities, each with its own difficulties. Let us discuss the typical difficulties for the three activities one-by-one.

(i) *Making a formal problem specification.* The nature of such problem specifications is that they remain close to the original process and, as a consequence, they are very detailed. So, the making of such a problem specification is usually much work. However, in many cases this large amount of work only occurs one time, since the changing of scenarios only requires a minor amount of work. In several cases, the work for the initial specification as well as for the later revisions may be diminished considerably by introducing standard modules in a parametrized form, which makes it possible to refrain from executing the specification process on the most detailed level. In Section 5 such a case will be treated in more detail. If such a higher level specification is not possible or if its effect is insufficient, then it may indeed be possible to lighten the burden of specifying many details by using rule-based methods. In Section 4, a case is described where this approach seems to be useful.

(ii) *Choosing a model type and specifying the model.* There is a strong relation between this activity and the next, since choosing the model type determines usually at least the type of algorithm to be applied. The choosing of the model type can sometimes profit from the introduction of rule-based selection methods which evaluate the characteristics of the problem specification: how important are the dynamics of the process, is it necessary to make a dynamic model and, if so, which dynamics have to be incorporated and what format should be chosen (the latter aspect points most directly at the type of algorithm to be used later). Also for specifying the model, rule-based methods can be of help, particularly if translation from problem specification to model requires some form of judgement. It may be necessary, as with the other subactivities, to combine rule-based choices with judgements by the user.

(iii) *Choosing an algorithm and executing it.* The selection of an algorithm for a given model format and, if necessary, the construction of this algorithm is related to the preceding question of choosing a model type, although the algorithm selection is more a technical question usually. This problem occurs, for example, if the model choice leads to some discrete optimization problem in a standard format for which some heuristic has to be constructed from standard components. Of course, it would be nicer to have one flexible algorithm available (like the simulated annealing in the example of approach **a**), but this is not always a feasible possibility. In fact, it may also be sensible to apply a rule-based procedure as search algorithm. A clear disadvantage would be its relative inefficiency and possibly a weak effectiveness, but an important advantage can be its flexibility.

4 Problem specification for manpower policy making.

For manpower scheduling problems it is quite often possible to apply a static approach. For manpower policy making, however, the dynamical characteristics are usually dominating and therefore most of the models used are essentially models which describe the natural process and the possibilities to bear influence on this natural process. The most important features of the models used are the level and type of detail and the degree to which the natural process can be affected. Therefore, the problem specification must contain the necessary information to choose those model features. Moreover, for manpower policy making, it is desirable to have the possibility of a scenario analysis as well as the possibility of letting the system complete a partially specified scenario in such a way that some goals are reached, if possible. With these remarks in mind it is useful to partition the problem specification in three parts:

a. The detailed state (a description of the current situation and of the process mechanism as far as it is considered as fixed);

b. The objectives (the description of the goals and constraints and their relative importance as far as relevant for the current problem);

c. The results of a model analysis (only to be filled in later).

For part **a**, one might use a kind of structure like used in Verhoeven [15] or in Verhoeven, Wijngaard, Wessels [16], which primarily categorizes the personnel and specifies the transition possibilities. For a problem specification one would use a detailed version of such a structure, categorizing at least over grade, grade age, age, location, education, sex. When translating to models there may be aggregation over some of these characteristics. Moreover, there may be given numbers or percentages for all or a part of the possible transitions. If career policies are considered as given, then they should be translated to such percentages. How important such percentages are for the process only becomes clear in part **b** of the problem specification. Also a partioning of categories in clusters may be given, together with basic data regarding salaries. The objectives of part **b** specify the goals and constraints of the user by determining, for instance, that the percentages of

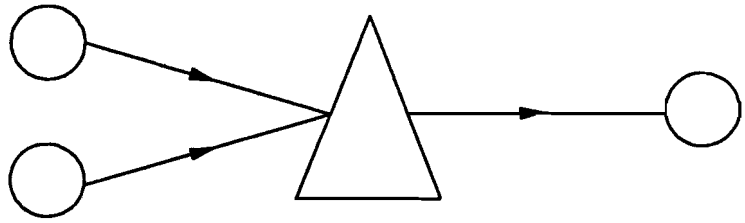

Figure 3: *A transition (denoted by a triangle) with two input places (circles) and one output place (circle).*

part **a** are either upper bounds or lower bounds or goals and also by determining the importance of not violating such bounds or of achieving such goals. Other objectives might be financial or specify numbers of personnel in different clusters in subsequent years. The number of objectives and also the lists of relative importances may by large. Actually, the numbers of data to be stored for part **a** and **b** are considerable and several of them have to be given by the incidental user, although the majority of the data can be made available for multiple use. Nevertheless, the burden on the user remains considerable. Therefore, it is sensible to look for means of facilitating the tasks of the user. Most of the data required from the user are trivial data which are quite obvious for the user, since they are related to other choices. Hence it is a natural approach to help the user with rule-based procedures which either fit details automatically, or prepare proposals for the user, or suggest that the user breaks some ties. The reactions of the user to proposals and the way the user breaks ties can be exploited in the rules for other data. A similar type of difficulty occurs during the step from problem specification to model for analysis. Here it is also useful to introduce rule-based procedures for the translation. Such rules may incidentally lead to a return to the problem specification phase for clearing-up some essential choice. For more information the reader is referred to Venema, Wessels [14], Van Vroonhoven-Van Beek [17] and Van Kraaij, Venema, Wessels [10], [11].

5 A language for specifying goods flow problems.

At a very elementary level, goods flow processes can be seen as being built with standard steps like translation, buffering, transformation. Actually, there is a strong analogy with distributed data processing and, therefore, one might expect that formal specification methods for distributed data processing might be useful for adaptation to specification methods for goods flow problems. As an example we take ExSpect (compare Van Hee, Somers, Voorhoeve [7]), a specification language which is based on coloured timed Petri nets. So, the Petri nets provide the generic model (for a general introduction to Petri nets and a recent overview, see Murata [12]; for coloured Petri nets, see Jensen [8]; for timed Petri nets see Zuberek [19]). A Petri net consists of transitions and places (to see the connection with logistics, the terms processors and channels would be better). Any transition is connected to one or more input places and to zero or more output places (see Figure 3).

At any moment in time there can be a set of tokens in any place. Each transition puts some weight to each of its input and output places and as soon as each of the input places

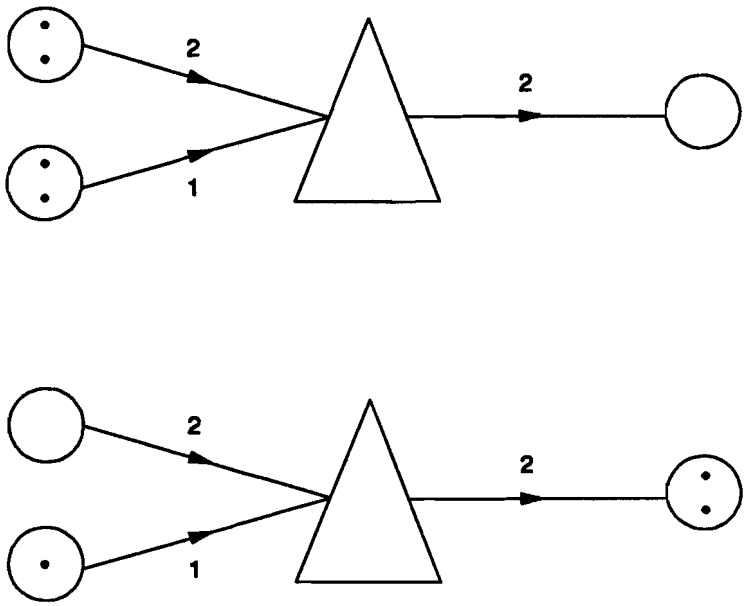

Figure 4: *The distribution of tokens just before and just after a firing in the case where one input place had one token more than necessary.*

of a transition has at least a number of tokens equal to its weight, then the transition can fire, meaning that it consumes from each of its input places a number of tokens equal to the weight of that place and produces a number of tokens for each of the output places equal to the weight of that place (see Figure 4).

Using these notions, one can easily describe simple goods flows. As an example, Figure 5 shows the description of an assembly process. The main missing aspect so far is the aspect of time, which can easily be brought in by providing any token with a time stamp and adding a delay to any transition. A transition can only consume tokens with an indicated time which has already elapsed and the new tokens for the output places get times equal to the firing time plus transition delay (see Figure 6, for an example of two parallel processes using the same tool).

The scope of application of this concept can be extended by giving the tokens a value or colour (apart from the time stamp), but we will refrain from this aspect for this exposition. Already from the simple illustrations in this paper it can be seen that this concept is well suited for specifying production and distribution processes. For more details on how to use this concept for specifying flexible manufacturing systems and logistic systems, we refer to Van der Aalst and Waltmans [2], [3] and for specifying scheduling problems to Carlier, Chretienne, Girault [6].

ExSpect provides a language based on Petri net models as mentioned. The user may formulate his specification in this language or use a graphic interface which provides the translation to the language. One of the nice aspects of this approach is that the specification may be seen as a prototype of the real system to be designed. This prototype may be used for experiments (simulations). So, ExSpect not only provides a language for making specifications, but it also takes care for making a simulation program based on the specification (ExSpect stands for Executable Specification Tool). A clear disadvantage of the approach so far is that specification of real systems leads to very extensive

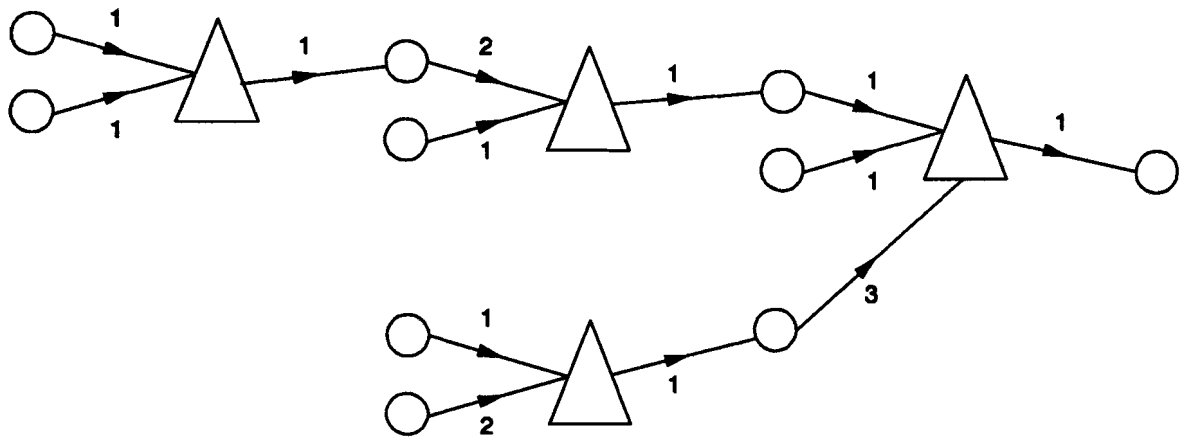

Figure 5: *An assembly process with six types of components coming from the outside and three types of subassemblies.*

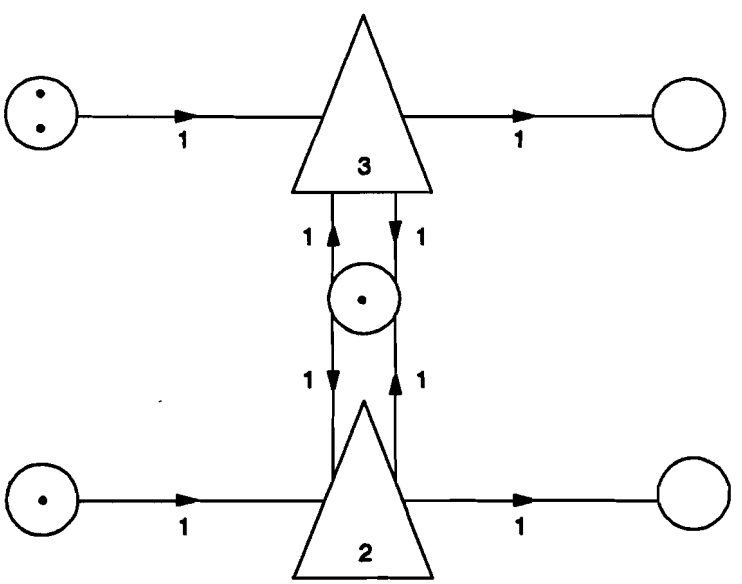

Figure 6: *Two parallel processes using the same tool. The transitions have delays of three and two time units respectively.*

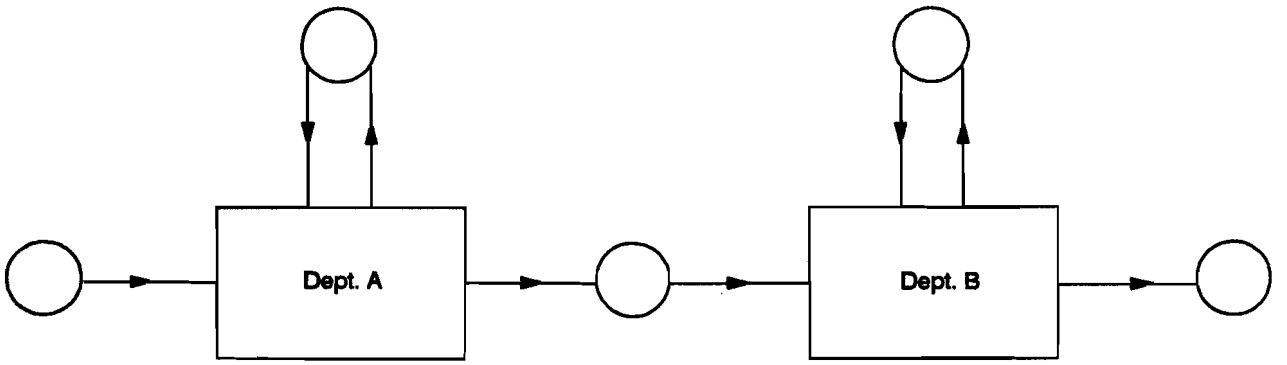

Figure 7: *A manufacturing unit consisting of two departments, where for each department an order may only be released if the number of active orders is below a given maximum, so the departments use workload control.*

descriptions which are difficult to handle for the user. Therefore, it was necessary to include a mechanism to construct specifications hierarchically and use standard modules as building blocks. In this way it becomes possible to get a goods flow specification on the screen where production departments are depicted as black boxes. On the other hand one can zoom-in on one of these black boxes and then see how the goods flow from one workstation to another proceeds. Zooming in on such a workstation, one gets a detailed view of the order processing at the workstation. This feature is illustrated in Figure 7, together with the fact that this way of specifications may integrate the specification of physical flow with the specification of the information processing and the control.

6 Comments and conclusions

The goal of this paper is not to present completely worked out approaches, but rather to show how a problem oriented way of thinking about decision support leads to R&D-problems of a completely different nature. It is also not implied that the approaches as presented in this paper are the right approaches for all sorts of problems. It is not even implied that these approaches give the best way for attacking the types of problems they were meant for. However, it should be clear from this paper that other research topics might be relevant than the ones usually propagated. To illustrate this point, we will discuss some of the problems mentioned earlier in more detail.

a. In combinatorial optimization there is a lot of interest in constructing heuristics for rather specific problems, whereas it might be at least as relevant to construct robust heuristics which are perhaps not the best for any specific type of problem, but are relatively good for a large variety of problems. Another approach might be to find ways for automatically constructing heuristics for specific problems.

b. Starting with problem oriented specifications like the one presented in sections 4 and 5, implies that the translation from specification to model for analysis might be a difficult exercise. Therefore, it is natural to try to construct methods of analysis which can be applied directly to problem specification. A production design problem, for

instance, may be essentially a queueing problem for which methods of analysis exist which are powerful as well as efficient. However, specialized OR-analysts underestimate systematically the difficulty of recognising the underlying queueing problem and of formulating the relevant model. The consequence of this argument is *not* that one should throw away the elegant mathematical models with their powerful and efficient methods of analysis. However, the consequence should be that one does not strive for a standard model at any price. It might be possible to do a sensible analysis directly on the problem specification or a close derivate, whereas the step to a standard model is non-trivial and the standard model does not allow a simple analysis. In **c** this will be worked out a little bit for the specification method of Section 5.

c. For goods flow problems many types of mathematical models are used intensively: linear programming models, combinatorial optimization models, simulation models, inventory models, queueing models. However, quite often the translation from problem specification to mathematical model is non-trivial and requires some essential decisions. On the other hand, it is possible to use a problem specification in ExSpect directly for a simulation. By its sheer nature, such a simulation on the most detailed model with a very general tool is much less efficient than a simulation with a less detailed model using a more dedicated simulation tool. However, if the simulation in ExSpect is practically feasible, then it provides a simple approach. Because of the inefficiency of simulation there are also attempts for more analytic approaches. For specificatons with deterministic times, as introduced in Section 5, it is indeed possible to give a time analysis (see Zuberek [19]), however, for the natural extension to stochastic times, the attempts did not lead to practically feasible approaches. As a compromise it is attempted to use time intervals rather than deterministic times or stochastic times and, indeed, the results are promising (see Van der Aalst [1]).

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, 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.

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

[4] J.F. Benders, J.A.E.E. van Nunen. A decision support system for location and allocation problems within a brewery. *Operations Research Proceedings 1981.* Springer, Berlin 1982, p. 96–105.

[5] J.W.M. Bertrand, J.C. Wortmann, J. Wijngaard. *Production control: a structural and design oriented approach.* Elsevier, Amsterdam 1990.

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

[7] K.M. van Hee, L.J. Somers, 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.

[8] K. Jensen. Coloured Petri nets. p. 248–299 in: W. Brauer, W. Reisig, G. Rozenberg (eds.), *Advances in Petri nets 1986 Part I: Petri nets, central models and their properties.* Springer (LNCS 254), Berlin 1987.

[9] L. Kleinrock. *Queueing Systems.* J. Wiley and Sons, New York 1975.

[10] M.W.I. van Kraaij, W.Z. Venema, J. Wessels. Automatic modelling and solving of strategic planning problems. *Methods of Operations Research* **63** (1990) 405–414.

[11] M.W.I. van Kraaij, W.Z. Venema. Support for problem solving in manpower planning problems. *Methods of Operations Research* (to appear).

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

[13] C.B. Tilanus. Operations Research in the Netherlands. *Europ. J. Oper. Res.* **18** (1984) 220–229.

[14] W.Z. Venema, J. Wessels. Automatic modeling and model handling for manpower planning systems. *Decision Support Systems* **4** (1988) 503–508.

[15] C.J. Verhoeven. *Techniques in corporate manpower planning; methods and applications.* Kluwer/Nijhoff Publishing, Boston 1982.

[16] C.J. Verhoeven, J. Wessels, J. Wijngaard. Computer-aided design of manpower policies. *Manpower Planning Reports* **16**. Technical University Eindhoven 1979.

[17] H.J.R.A.J. Van Vroonhoven-Van Beek. The data handling component of the decision support system Formasy. Internal report, Technical University Eindhoven 1990.

[18] W.H.M. Zijm. Operational control of automatic PCB assembly lines. p. 146–164 in: G. Fandel, G. Zäpfel (eds.), *Modern Production Concepts.* Springer, Berlin 1991.

[19] W.M. Zuberek. M-timed Petri nets, priorities, preemptions and performance evaluation of systems. p. 478–498 in: G. Rozenberg (ed.), *Advances in Petri nets 1985.* Springer (LNCS 222), New York 1986.

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.