Detecting Non-Ergodic Simulation Models of Logistics Networks[*]

Falko Bause
LS Informatik IV
Universität Dortmund
D-44221 Dortmund
falko.bause@udo.edu

Jan Kriege
LS Informatik IV
Universität Dortmund
D-44221 Dortmund
jan.kriege@udo.edu

ABSTRACT

Simulation is a frequently applied method when analysing logistics networks. Also within the Collaborative Research Center 559 "Modelling of Large Logistics Networks" simulation is broadly applied and process chains are used as a mutual basis for model development and description. Previous research activities exposed non-ergodicity of models as one of the typical application-specific problems which are difficult to discover by simulation.

In order to detect non-ergodic models the problem has been reduced to its core employing the more analysis oriented modelling formalism of Petri nets. With the help of the Petri net formalism we developed an efficient method for the detection of non-ergodic models. Since Petri nets is not the common modelling paradigm for logisticians, this method had to be made available in the process chain modelling world of the logistics area, additionally supported by an appropriate tool.

This paper describes our corresponding approach and also demonstrates the process of identifying a problem class in an application area, reducing it to its core, establishing a solution in an analysis-oriented formalism and making corresponding techniques available in the application-oriented modelling world and thus also available for the end-user.

Categories and Subject Descriptors

I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis

General Terms

Performance

Keywords

Logistics, Simulation, Ergodicity, Process Chains, Petri Nets

[*]This research was supported by the Deutsche Forschungsgemeinschaft as part of the Collaborative Research Center "Modelling of Large Logistics Networks"(559).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Valuetools '07, October 23-25, 2007, Nantes, France
Copyright 2007 ICST 978-963-9799-00-4.

1. INTRODUCTION

Process chains are often used for the modelling of logistics networks (cf. [19, 24]). Most process chain paradigms are informal or at most semi-formal and cannot be directly used for the analysis of quantitative aspects. The Collaborative Research Center 559 "Modelling of Large Logistics Networks" (CRC 559; [25]) is a research project consisting of several sub-projects combining the different competences of the field of logistics: engineering, economics, statistics and computer science. The CRC 559 aims at handling the complexities of large logistics networks by model based analysis using the process chain paradigm of [19, 20] as a basic principle. The need for a common formalism for the modelling of logistics networks, that allows for automatic mappings to analysis techniques, resulted in the *ProC/B*-paradigm ([6]), which reduces the gap between informal process chain descriptions and formal models. Modelling in *ProC/B* is supported by the *ProC/B*-toolset ([5]), which allows a, mostly graphical, description and analysis of *ProC/B*-models.

Due to size and complexity, *ProC/B*-models are often analysed by means of simulation. For some models the simulation showed surprising results, that are not immediately understandable (cf. Fig. 3): The results seem stable for some period of time, but after a longer observation time the run changes to strange behaviour. Individual inspection of such models showed, that these results are caused by non-ergodicity and that this non-ergodic behaviour is caused by the structure of the model and cannot be avoided by a change of model parameters like for example interarrival times. Non-ergodicity essentially means that a steady-state distribution does not exist (cf. [13, 21]), so that nonterminating simulations are useless. In most cases non-ergodicity of the model hints at an incorrect modelling of the system, indicating that specific characteristics have been neglected or misrepresented.

Since in most cases it takes long and expensive runs to discover such situations by simulation, it is desirable to detect non-ergodic models before the simulation is started. A method for detecting such situations has been found for Petri nets ([2]). This paper describes how the method can be applied to *ProC/B*-models and how it can be integrated into the *ProC/B*-toolset. Petri nets are a formalism well known for verifying properties of a model. In [27, 28] for example Petri nets are used to formalise event-driven process chains for the description of business processes. While most modelling and simulation tools support the user in checking the syntactical correctness of the model (cf. [16] for example), to the best of our knowledge a detection of non-ergodic models prior to the simulation is not included in any simulation software.

This paper is organised as follows: In Sect. 2 the *ProC/B*-paradigm is introduced. Sect. 3 gives examples for non-ergodic models of logistics networks and in Sect. 4 a Petri net-based technique for detecting such models is presented. Sect. 5 describes the integration of the proposed technique into the *ProC/B*-toolset and in Sect. 6 concluding remarks are given.

2. THE PROC/B-PARADIGM FOR MODELLING LOGISTICS NETWORKS

The process chain paradigm [19, 20] is used as a common modelling language within the CRC 559 offering experts from different areas a mutual basis for communication and model development. Since it is an informal modelling paradigm, it offers on the one hand flexibility when describing systems, but on the other hand makes analysis difficult. Usually the analyst has to transform the process chain description manually into a suitable input of some simulation tool. This transformation activity typically includes adding further details and specifying informal descriptions more precisely. In addition to extra work for the analyst, the transformation also raises a consistency problem, since it is difficult to decide whether the simulation model expresses the intention of the informal model. In order to diminish these problems, parts of the process chain paradigm were enhanced and stated more precisely [5]. The resultant modelling paradigm is called *ProC/B*. *ProC/B* accounts for the specifics of the application area by capturing the structural hierarchy of logistics networks in form of functional units and the behavioural hierarchy by process chains (PCs) whose activities can be refined into sub-activities which may be further refined etc. *ProC/B* combines these two hierarchies in one description. Function units (FUs) might offer services, which can be used by activities of process chains. Each service is again described by a process chain.

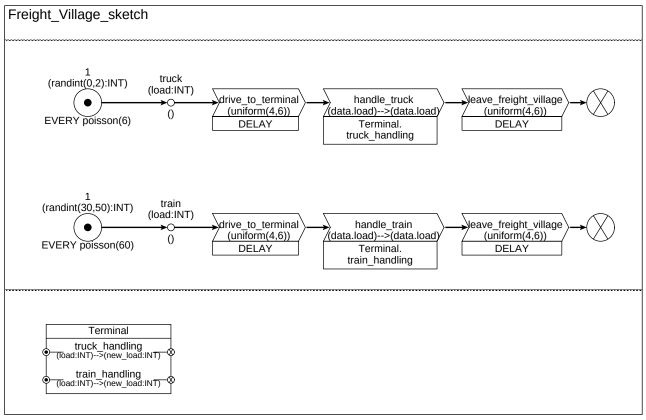


Figure 1: Freight Village

Figs. 1 and 2 present an example of a *ProC/B*-model representing a (simplified) freight village. The top level of the model (see Fig. 1) is specified by FU `Freight_Village_sketch` whose behavioural part is described by two PCs: `truck` and `train`. The structure part consists of a single (user

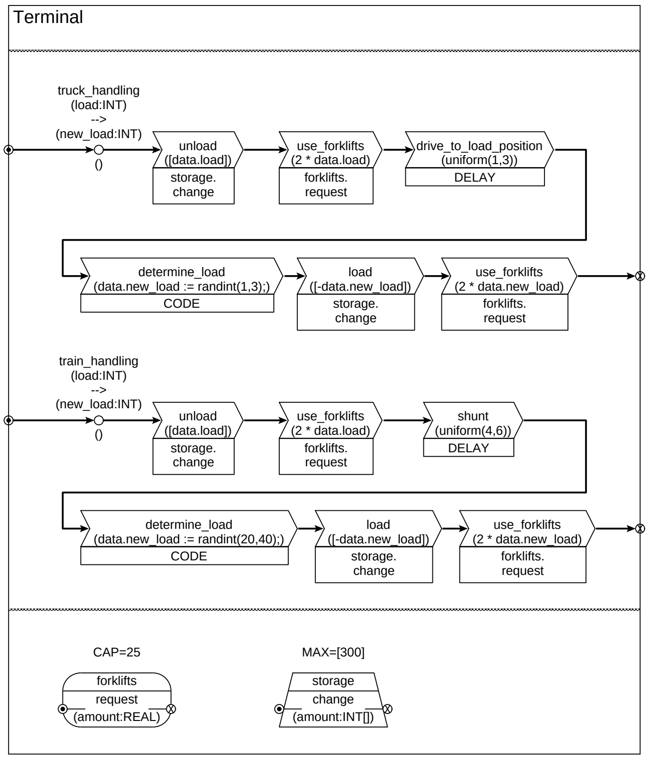


Figure 2: Terminal of Freight Village

defined) FU, named `Terminal`, which offers two services: `truck_handling` and `train_handling`. Services can be compared to functions in programming languages. In the shown example both services have an input parameter (`load`) and an output parameter (`new_load`). Behaviour and structure part of a FU specification are interrelated by expressing which service of which FU performs an activity. In Fig. 1 the two PCs `truck` and `train` consist of three process chain elements (PCEs) each, and in both cases the second activity calls a service of FU `Terminal`. The inner view of FU `Terminal` is shown in Fig. 2. The offered services are specified by PCs and some of their activities use the services of two so-called standard function units which offer predefined services (`request` and `change`). *ProC/B* offers two kinds of standard FUs: servers and counters. Servers (see `forklifts` in Fig. 2) capture the familiar behaviour of traditional queues and counters (see `storage` in Fig. 2) support the manipulation of passive resources. A change to a counter is immediately granted iff its result respects specified upper and lower bounds; otherwise the requesting process gets blocked until the change becomes possible.

One of the advantages of *ProC/B* and process chains in general is the visualisation of behaviour, which supports the communication between experts of an application area. So the freight village model of Figs. 1 and 2 reads as follows: Processes for process chain `truck` are generated according to a Poisson distribution (with a mean of 6 time units). Each truck has a load which is initially chosen by random according to an uniform distribution (0, 1 or 2). After incarnation, the truck "drives" to the terminal which is modelled here by a delay of the process for a uniformly distributed duration. Afterwards the truck "is handled" by service `truck_handling` of `Terminal`. This might result in a change of the truck's load. Finally the truck "leaves" the freight village and the process terminates at the sink. Considering Fig. 2 we see that handling a truck means first to unload the truck, which is possible if the `storage`'s capacity of 300 units is not exceeded. Afterwards the server `forklifts` is called, which is a multi-server queue with 25 servers and a (default) FIFO service strategy. The service time for the requesting process is determined by the expression `2 * data.load` thus modelling the time for unloading a loaded truck. (Remark: Access notations to parameters and variables of processes are prefixed with keyword *data* for technical reasons in order to distinguish them from global variables. Global variables are not shown in Figs. 1 and 2.) Afterwards the truck "drives" to a new position (which is again modelled by a delay of the process) and determines the new load. The new load is removed from the storage if possible (the default lower bound of a counter is 0) and finally the service `request` of the server `forklifts` is called again before the process "leaves" the terminal. The behaviour of `train` and service `train_handling` reads similarly.

As one might imagine, it is possible to specify *ProC/B*-models precisely enough that an automatic analysis becomes possible. The formerly manual transformation process can now be automated thus avoiding the mentioned consistency problem. Within the CRC 559 a corresponding toolset has been developed which provides a graphical user interface to specify *ProC/B*-models and transformer modules which map *ProC/B*-models to the input languages of existing analysis tools. For more details on *ProC/B* and the corresponding toolset we refer the reader to [5, 6].

Not surprisingly, simulation is often applied for a detailed analysis, since it is applicable to all *ProC/B*-models. In most cases the logistician is interested in the steady-state behaviour of the system/model, e.g. in long-term mean values, but simulation has the disadvantage of providing only statistical results. In the early beginnings of the CRC 559 an interesting effect was discovered [3]: Several models of logistics networks (also the simple model of Figs. 1 and 2) show non-ergodic behaviour, implying that the corresponding steady-state mean values do not exist. Non-ergodicity per se is not a surprising effect, since overload situations are often encountered when determining the model's peak performance. Usually an appropriate choice of the parameters results in an ergodic model for which steady-state performance figures can be determined. But in the domain of logistics networks one finds typical situations where non-ergodicity can not be avoided by adjusting parameters. For those models it turns out that non-ergodicity is an intrinsic characteristic of the model.

3. NON-ERGODICITY IN MODELS OF LOGISTICS NETWORKS

The freight village model presented in Sect. 2 is a variant of the model considered in [3]. Only some parameters have been changed to reduce the simulation effort. Fig. 3 depicts a possible simulation result. A point at model time t of the shown curve denotes the mean number of trains at FU `Terminal` for the time interval $[0, t]$. Normally one would expect convergence to some limit value. The first part of the curve of Fig. 3 suggests convergence.

Inspecting the simulation result in detail, one realizes that the confidence interval width becomes very small, e.g. at time 16000 the 95% confidence interval is about $2.4 \pm 2\%$. Usually automatic stopping rules would have terminated the simulation at that point in time or even earlier, so that we would not have seen the second part of the curve which hints at non-ergodic behaviour.

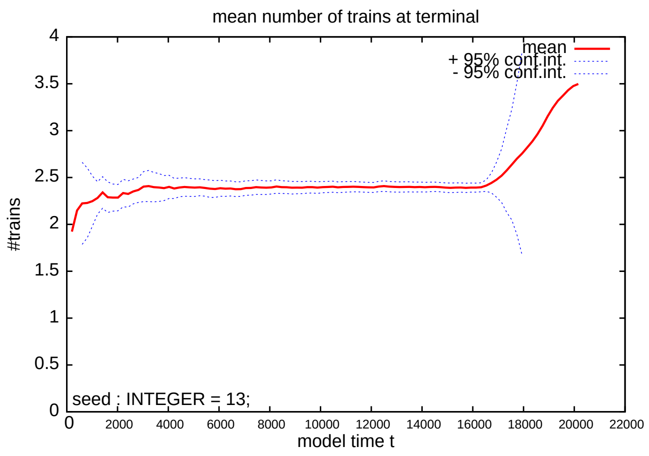


Figure 3: A possible simulation result of the freight village model from Figs. 1 and 2

[3] also presents an explanation for this behaviour. Informally explained, non-ergodicity is caused by the interdependence of trains and trucks: Both type of vehicles deliver and pickup load units, which are temporarily stored in the storage. Both, trucks and trains, are blocked whenever delivery or pickup is not possible, because the storage is empty or full. In a way, loaded vehicles can be interpreted as being the server for unloaded vehicles and vice versa. Even though the average number of delivered load units matches the average number of loaded units, the model is non-ergodic due to stochastic effects (cf. the specification of arrival streams, new load units and driving times). The following Markov chain explains this kind of non-ergodic behaviour in a more formal manner.

Assume that the storage capacity is n and that we have only two type of carriers: type V1 carriers which are loaded by one unit each and are going to deliver their load to the storage, and type V2 carriers which are unloaded and try to load one unit from the storage. Further assume that the corresponding arrival processes are determined by Poisson streams with rate λ for type V1 and μ for V2. Let the chain's state space consists of all integers

$0, 1, \dots, n$	for $0, 1, \dots, n$ storage spaces filled and no carriers waiting
$n+1, n+2, \dots, \infty$	for n storage spaces filled and $1, 2, \dots$ type V1 carriers waiting
$-1, -2, \dots, -\infty$	for 0 storage spaces filled and $1, 2, \dots$ type V2 carriers waiting

If we define that driving, shunting, and using the forklifts are immediate, i.e. timeless activities, we get the Markov chain (of tangible states) shown in Fig. 4. The global balance equations for this Markov chain have the solution

$$\pi_i = \rho^i * \pi_0, \qquad i = -\infty, \dots, \infty \quad \text{with} \quad \rho := \frac{\lambda}{\mu}$$

showing that π can not be normalised to a probability distribution, i.e. $\sum_{i=-\infty}^{\infty} \pi_i = 1$ can not be satisfied. Of course, the Markov model is very simple, but shows an interesting characteristic: Non-ergodicity can not be "fixed" by selecting appropriate values for λ and μ, since it is caused by the structure of the chain.

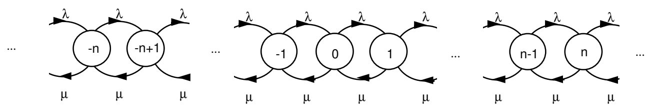


Figure 4: Non-ergodic Markov chain

Also [3] reports about several simulation runs of the freight village model using different parameter settings. In the end all simulations showed non-ergodic behaviour. Several other models of logistics networks turned out to be "structurally non-ergodic" as well and specific parts of such models have been identified being responsible for non-ergodicity. E.g., stock-keeping activities (cf. Fig. 7) and typical assembly or packing operations might lead to non-ergodic behaviour (cf. [2]). In most cases non-ergodicity of the *ProC/B*-model hints at an incorrect modelling of the system. Considering the freight village model, e.g., we ignored existing time tables for trains and control processes managing delivery schedules which would have avoided non-ergodicity in this case. Having Fig. 3 in mind, it is desirable to detect such incorrect models before starting the simulation, also remembering that we presumably have stopped the simulation of the freight village model before the effect shows up. In our example non-ergodicity was caused by the model's structure, thus it seems an obvious idea to investigate this structure, particularly having in mind the usually large complexity of state-based analysis techniques. In order to develop new techniques it is often beneficial to select less complex modelling formalisms. Analysis based on the inspection of the model's structure is, e.g., the domain of Petri Nets [8, 22].

4. A PETRI NET-BASED TECHNIQUE FOR DETECTING NON-ERGODIC MODELS

In [4] a simple Generalized Stochastic Petri Net (GSPN) structure is presented (see Fig. 5), which exhibits the same effect as the freight village model of Figs. 1 and 2. The GSPN is non-ergodic, since its Markov chain (of tangible states) is the same as depicted in Fig. 4. Note that all timed transitions of a GSPN have, by definition, an exponentially distributed firing delay and that for the net of Fig. 5 we have $M(p_1) > 0$ implying $M(p_2) = 0$ and $M(p_2) > 0$ implying $M(p_1) = 0$, so that $i := M(p_1) - M(p_2)$ is a suitable state descriptor for tangible states.

Since our interest is in efficient techniques, the question is whether one can detect such nets without inspecting the state space. Ergodicity in bounded, i.e. finite-state (Generalized) Stochastic Petri is related to the existence of home states and several authors have investigated this property, very often in the context of special net classes, see e.g. [8, 9, 11, 26]. In case of infinite-state GSPNs the existence of home states does not characterise ergodicity anymore. It is still a necessary condition, since it corresponds to the irreducibility of (a subset of) the corresponding Markov chain, but sufficiency does not hold as shown by the example in Fig. 4.

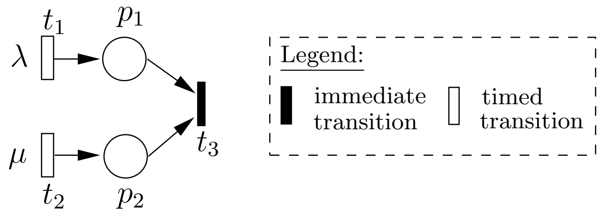


Figure 5: A non-ergodic GSPN $\forall \lambda, \mu \in \mathbb{R}^+$ (cf. [4])

In the following we will denote with C the incidence matrix of the net (cf. [8]). The i-th column of C describes the effect on the marking by firing transition t_i. E.g., the incidence matrix of the GSPN of Fig. 5 is $C = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{pmatrix}$. Let $m \in \mathbb{N}$ denote the number of transitions of the GSPN. The kernel of matrix C is defined by $kernel(C) := \{x \in \mathbb{R}^m \mid C \times x = 0\}$.

Let $N(z)$ be a vector counting the number of transition firings in the time interval $[0, z]$. The mean firing flow vector $N \in \mathbb{R}^m$ is then defined by $N := \lim_{z \to \infty} E[N(z)]/z$ where $E[\quad]$ denotes the expectation operator.

Assuming an ergodic GSPN, it is obvious that the mean firing flow vector N exists and that furthermore the expected input flow of tokens at a place equals its expected output flow (cf. [2, 14]). The last fact can be expressed using the incidence matrix C of the GSPN giving

$$C \times N = 0 \qquad (1)$$

How can we exploit this equation for detecting non-ergodic nets? Usually the determination of N is difficult, but for some transitions we can identify the corresponding components easily. Consider our example of Fig. 5 again. Obviously, the components n_1, n_2 of the mean firing flow vector $N = (n_1, n_2, n_3)$ are directly given by definition, because t_1 and t_2 are source transitions, i.e. $n_1 = \lambda, n_2 = \mu$. Furthermore, from Eq. (1) we know that N is in the kernel of matrix C. A basis for the kernel of the incidence matrix of the net given in Fig. 5 is $(1, 1, 1)$ showing that $\lambda = \mu$ has to hold, if the GSPN is assumed to be ergodic. This shows that the net is sensitive towards even small changes of the firing rates and therefore a critical candidate. Theoretically the net might be ergodic for $\lambda = \mu$ (for our example we know that it is non-ergodic, but this can not be detected from Eq. (1) alone), but especially when using simulation for analysis we might run into a serious problem. The reason is that, due to the finite number representation in computers, we might not run the original model, but rather simulate a (slightly) perturbed, and thus non-ergodic model [23].

Determining information on some components of N is not only possible for source transitions, but also for transitions which partially exhibit an Equal-Conflict (PEC) net structure. Conflicting transitions in an Equal-Conflict net exhibit the important property that at any marking all conflicting transitions or none of them are enabled [26]. An example of such a net structure is shown in Fig. 6. Because of the net structure, we can deduce that $n_2/n_3 = \lambda_2/\lambda_3, n_2/n_4 = \lambda_2/\lambda_4$ and $n_3/n_4 = \lambda_3/\lambda_4$. Furthermore a basis for the kernel of the incidence matrix C is given by $\{(1,0,1,0,1),(2,1,0,1,1)\}$ implying $n_2 = n_4$ and thus $\lambda_2 = \lambda_4$. The last fact shows that the net of Fig. 6 is sensitive towards changes of the firing rates for transitions t_2 and t_4.

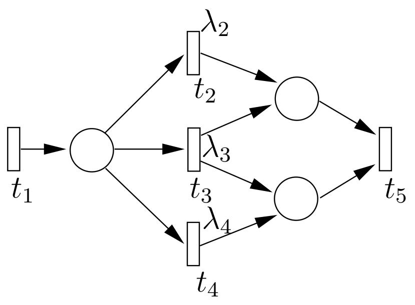


Figure 6: Transitions t_2, t_3, t_4 in PEC

In [2] this kind of sensitivity is called *e-sensitivity*, since it indicates potential non-ergodic nets. Also in [2] a sufficient criterion for detection of e-sensitive nets is presented. The criterion is based on a rank condition which can be efficiently tested. E.g., for the net of Fig. 6 one only has to take the components for transitions t_2, t_3, t_4 into account and since

$$\begin{aligned} & \text{rank}(\quad ((0,0,1,0,0)^{tr}(0,1,0,1,0)^{tr}) \quad) \\ = \quad & 2 < 3 = |\{t_2,t_3,t_4\}| \end{aligned}$$

we know that the firing rates are dependent on each other, showing that the GSPN is e-sensitive.

More precisely stated one can detect e-sensitive GSPNs by finding a PEC set $\tilde{T}$ for which

$$\text{rank}\left(\left(Proj(k_1,\tilde{T}) \ldots Proj(k_r,\tilde{T})\right)\right) < |\tilde{T}| \qquad (2)$$

holds, where $k_1, \ldots, k_r$ is a basis of kernel(C) and $Proj$ denotes the projection of vector k_i onto $\tilde{T}$. Note that all transitions of a PEC set need to be of the same kind, either timed or immediate, since otherwise the GSPN is not live. Rank condition (2) can be checked efficiently, since only maximal PEC sets need to be investigated, cf. [2].

Employing this rank condition shows that typical situation in logistics networks might lead to non-ergodic models when not modelled carefully. Fig. 7 shows the main activities when using a storage. Assuming that delivery and pickup is performed by different type of carriers and consequently modelled by different "arrival streams", the net of Fig. 7 shows a customary model. Unfortunately, the firing rates of the two source transitions are dependent or in other words the net is e-sensitive, since the kernel of the incidence matrix is one dimensional.

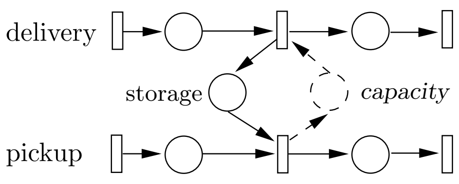


Figure 7: Stock-keeping scenario

The rank condition also identifies our freight village example from Sect. 2 as an e-sensitive GSPN, indicating non-ergodicity. Fig. 8 depicts the core behaviour of the terminal using averages for loading and unloading volumes.

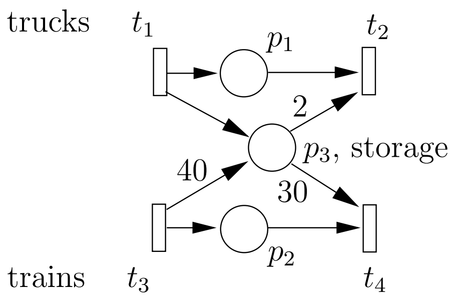


Figure 8: GSPN for freight village

The GSPN has two source transitions, i.e. two transitions in PEC, namely t_1 and t_3. Calculating the kernel of the incidence matrix gives the base vector $(10,10,1,1)$ showing a dependence of the firing rates of the two source transitions, since the rank (here 1) is less than the number of elements in the PEC set (here 2).

The former examples show that the rank condition might help identifying non-ergodic models. The simplicity of the modelling formalism, in our case Petri nets, definitely helped a lot when developing the described technique. Unfortunately, from the point of view of the logistician, which is our intended end-user, Petri nets are not an adequate modelling formalism for describing logistics networks. Far from it! We are already happy about the situation that the logistician accepts the more formal version of process chains, i.e. *ProC/B*.

So the question comes up how to make the rank condition, being formulated for Petri nets, available for the detection of non-ergodic *ProC/B*-models. Generally one has two possibilities: one is to adapt the Petri net technique for *ProC/B*, the other is to map *ProC/B*-models to adequate Petri nets. We selected the second possibility, since it additionally offers the chance to use further Petri net based techniques.

5. TOOL SUPPORT

To apply the technique described in Sect. 4 to *ProC/B*-models, the process chain model has to be transformed to a Petri net. This section shows how the transformation can be performed and how the technique can be used to test *ProC/B*-models for non-ergodicity. The test for detection of non-ergodic models has been implemented within the *ProC/B*-editor, additionally the Petri net can be exported to the APNN-format ([7]) making further techniques available being provided by the APNN-toolbox ([10]).

To use the test for non-ergodicity for *ProC/B*-models basically three steps have to be performed: First the *ProC/B*-model has to be transformed to a Petri net, before in a second step the test for non-ergodicity can be accomplished. Finally the results of the test have to be retranslated to the *ProC/B*-model.

The toolset maps *ProC/B*-models to hierarchical coloured Petri nets ([17]). Function units are represented by substitution places. Tokens are used to represent processes,

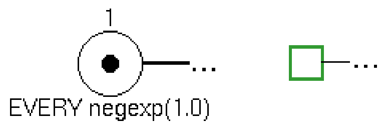


Figure 9: Mapping from *ProC/B* to Petri nets: Source *(green rectangle represents timed transition)*

while colours are used to distinguish between different process chains. The different elements of a *ProC/B*-model are mapped separately to Petri net constructs and connected afterwards. Some *ProC/B*-elements and their mappings to Petri net constructs are shown in Figs. 9-12.

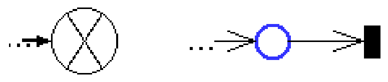


Figure 10: Mapping from *ProC/B* to Petri nets: Sink

A source, that generates processes in a *ProC/B*-model is mapped to a transition, that will generate tokens. Since a source is the starting point of a process chain it only has an outgoing arc to which the rest of the process chain is connected (depicted in Fig. 9 by three dots). Each source will generate tokens of a different colour. When mapping the services of function units, that are used by several different process chains, the colours are needed to distinguish between process chains, such that after service call only that process continues which initiated the call. A sink as an endpoint of

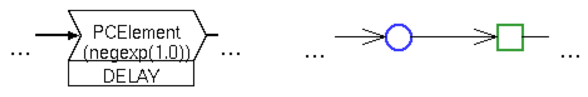


Figure 11: Mapping from *ProC/B* to Petri nets: Process-Chain-Element (PCE)

a process chain has only an incoming arc and is mapped to a place to which the rest of the Petri net is connected, and an immediate transition, which destroys tokens and thus terminates processes (see Fig. 10).
A PCE can be transformed in different ways, depending on whether it is used as a Delay-PCE or for calling a service of a FU. A Delay-PCE and its transformation is shown in Fig. 11. The transformation of a PCE connected to a FU (shown in Figs. 13 and 14) will be explained later. A Delay-PCE is mapped to a place and a timed transition, which denotes the amount of time the PCE consumes.

Finally Fig. 12 shows the mapping of an And-Connector as an example for the different connector-types available in *ProC/B*. An opening And-Connector splits a process chain in two or more parallel branches, that are merged again by the closing And-Connector. The Petri net representation consists of a transition that is connected to several places (one for each branch) and will create a token on each of those places. The closing connector is mapped to a transition that will destroy one token from each branch.

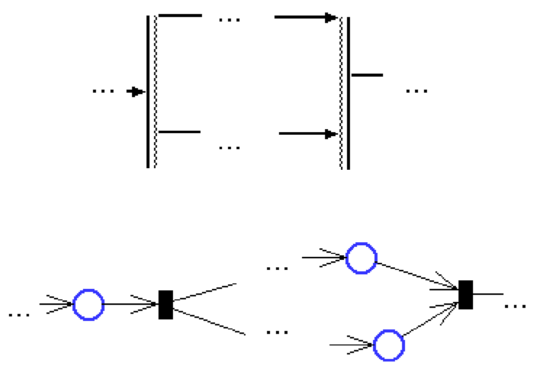


Figure 12: Mapping from *ProC/B* to Petri nets: And-Connector

Figs. 13 and 14 show the Petri net transformation of the freight village model from Sect. 2. Fig. 13 shows the top-level of the model. The function unit `Terminal` is translated to a substitution place. For each service of the FU, places and transitions are created where the PCEs that use the service are connected. The inner net of the substitution place is shown in Fig. 14, containing the mapping of the two services of the function unit. Please note that the names of the places and transitions have been taken from the elements of the *ProC/B*-model and may not be unique, since the mapping of a *ProC/B*-element may result in several places and transitions. Though all of the places and transitions have an additional unique ID not displayed in the figures. The two figures show the Petri net representation of a PCE connected to a service of a function unit. Depending on the type of the FU, this representation will look different: In Fig. 13 two PCEs are connected with the `Terminal`, Fig. 14 shows the connection of PCEs and counter `storage`, which is represented by a single place. The function unit `Terminal` is represented by a substitution place. The transitions connected to the substitution place are sockets. Each of those transitions has a counterpart within the net contained in the substitution place, the so-called port (see [15] for a detailed description of substitution places). For each PCE, that uses the service of a function unit, a unique colour is assigned to the transitions and places, that are created during the transformation of the service, to make sure, that after returning from the service call, the process continues at the correct place. In order to map an access to a counter correctly the user has to specify average values for the loading and unloading volumes, since in most cases those volumes are not fixed values but taken from a probability distribution (cf. Sect. 4 and Fig. 8).
It should be noted, that the transformation of the *ProC/B*-model to a Petri net underlies some limitations: The *ProC/B*-model may contain variables of different data types like integer, real or boolean, that can hardly be expressed by a Petri net and are ignored during the transformation. Some parts of the model, that access those variables, like for example branching conditions at a connector, cannot be mapped exactly to the Petri net. So basically the created Petri net reflects the structure of the *ProC/B*-model, while the be-

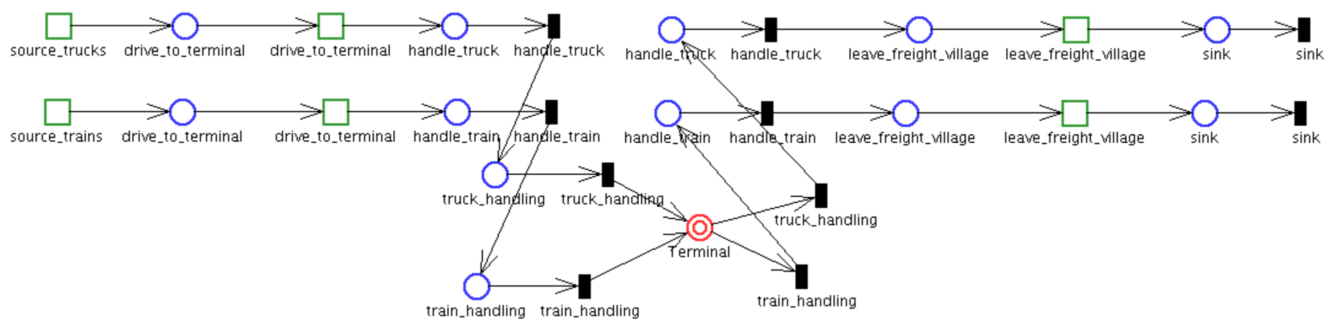


Figure 13: Petri net transformation of the freight village from Sect. 2

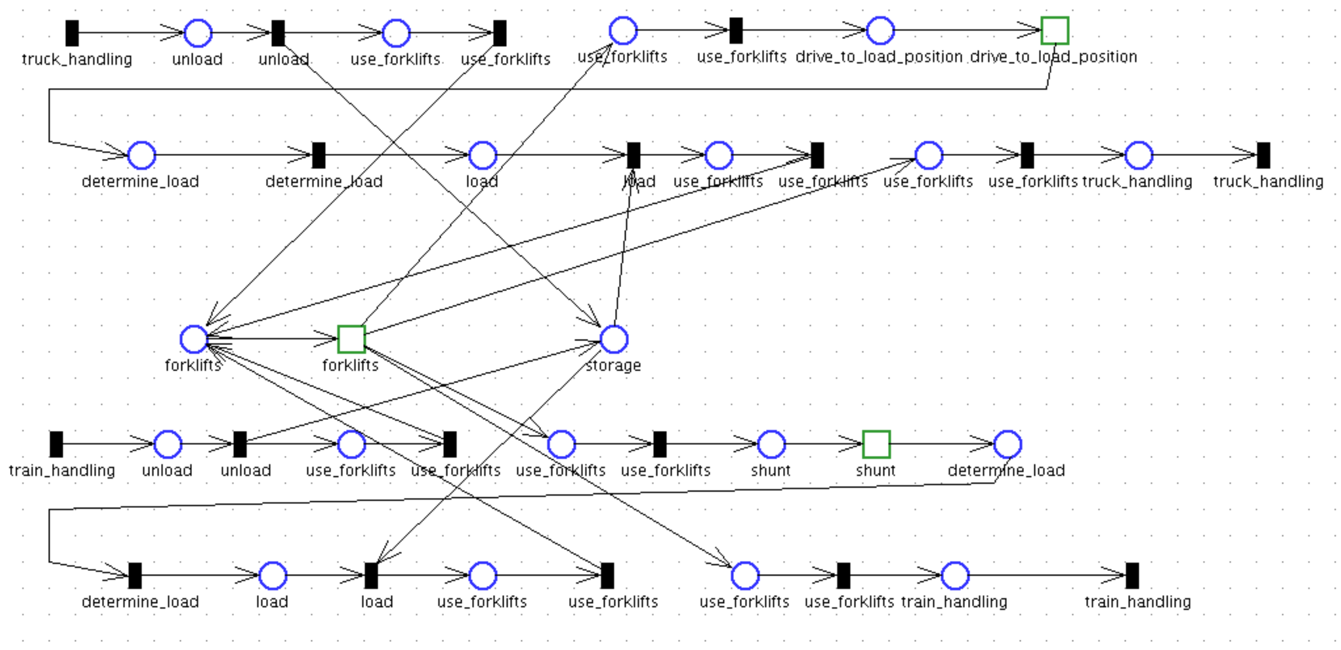


Figure 14: Petri net transformation of the function unit `Terminal` of the freight village from Sect. 2

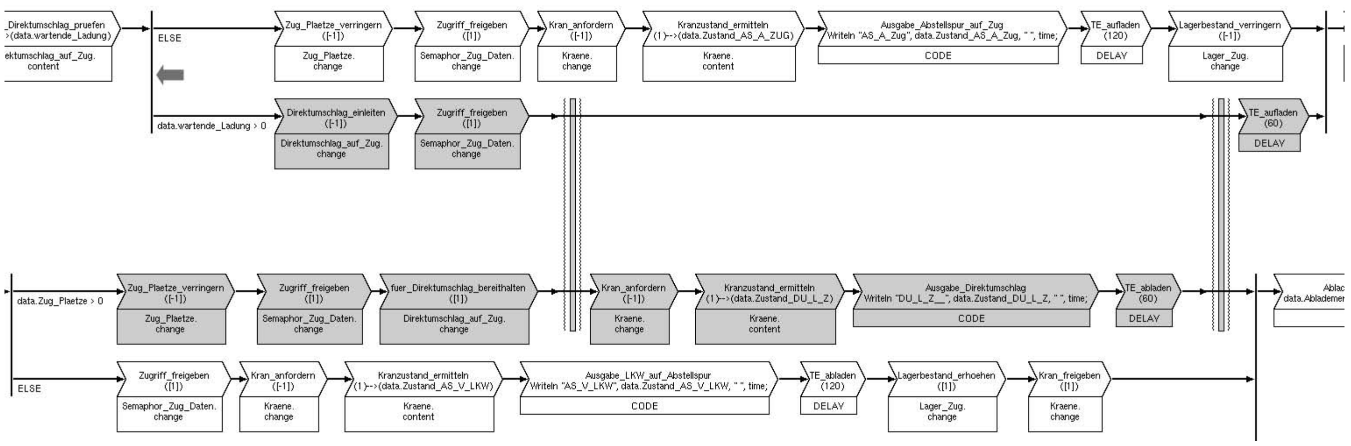


Figure 15: Results from non-ergodicity test

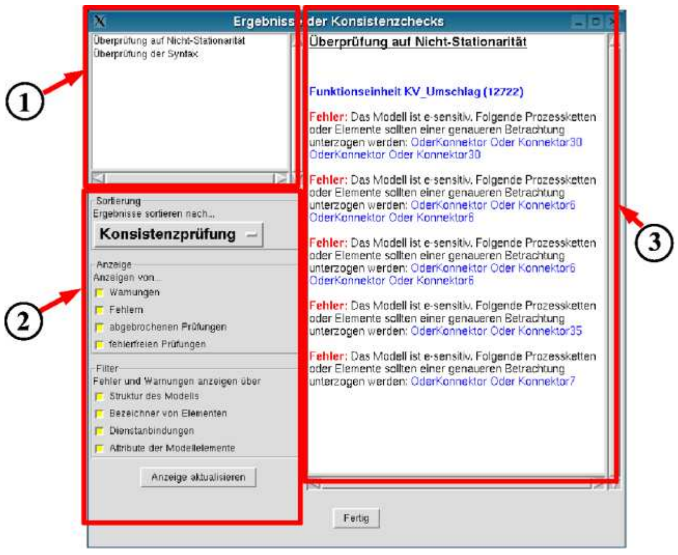
haviour may not be completely identical. This might result in so-called non-faults [1], which denote characteristics of the Petri net (e.g. a non-live net) which are not consistent with the *ProC/B*-model (e.g. the original model might be live due to boolean expressions at OR-branches). The problem of non-faults can not be avoided when abstracting. Thus the analysis based on Petri nets gives further results usually not obtained with such a certainty from a simulation, but still requires some insight from the user.

After the *ProC/B*-model has been translated to a Petri net the test from Sect. 4 can be performed. In case non-ergodicity is detected, one or more transitions are identified, that are in PEC and for which the cardinality of the PEC-set is greater than the rank of the matrix consisting of projected base vectors (cf. Cond. (2)). These results have to be retranslated to the *ProC/B*-model to be helpful for the modeller. That's why the relation between places and transitions of the Petri net and the elements of the *ProC/B*-model are stored, while the mapping from the *ProC/B*-model to a Petri net is done. Note, that in most cases a *ProC/B*-element is linked to several places and transitions, but each place and transition is linked to exactly one *ProC/B*-element. Therefore the corresponding *ProC/B*-element for a transition having been identified by the test, can be determined unambiguously.

The mapping from *ProC/B* to a Petri net, the test for non-ergodicity and the mapping of the results from the Petri net back to the *ProC/B*-model have been integrated into the *ProC/B*-toolset ([18]). The test can be selected for the whole model or only parts of it. Details of the test are hidden, so that modellers need not be familiar with Petri nets.

The outcome of the test is presented in a result window together with the results from another consistency check that performs a syntax check. The result window consists of three parts as shown in Fig. 16: The selection list contains all FUs for which errors have been found. Depending on the options selected, some of the messages can be suppressed. The remaining messages will be displayed in a sorted manner in the right part of the result window. The result messages are composed of a description of the error and the name of the affected element being presented in a HTML-like fashion. Clicking on one of the element names in those error messages opens a window of the associated function unit and highlights the element.

As already mentioned the test identifies the model of



Überprüfung auf Nicht-Stationarität
Überprüfung der Syntax

Sortierung
Ergebnisse sortieren nach...
Konsistenzprüfung

Anzeige
Anzeigen von...
Warnungen
Fehlern
abgebrochenen Prüfungen
fehlerfreien Prüfungen

Filter
Fehler und Warnungen anzeigen über
Struktur des Modells
Bezeichner von Elementen
Dienstanbindungen
Attribute der Modellelemente

Anzeige aktualisieren

Überprüfung auf Nicht-Stationarität

Funktionseinheit KV_Umschlag (12722)

Fehler: Das Modell ist e-sensitiv. Folgende Prozessketten oder Elemente sollten einer genaueren Betrachtung unterzogen werden: OderKonnektor Oder Konnektor30 OderKonnektor Oder Konnektor30

Fehler: Das Modell ist e-sensitiv. Folgende Prozessketten oder Elemente sollten einer genaueren Betrachtung unterzogen werden: OderKonnektor Oder Konnektor6 OderKonnektor Oder Konnektor6

Fehler: Das Modell ist e-sensitiv. Folgende Prozessketten oder Elemente sollten einer genaueren Betrachtung unterzogen werden: OderKonnektor Oder Konnektor6 OderKonnektor Oder Konnektor6

Fehler: Das Modell ist e-sensitiv. Folgende Prozessketten oder Elemente sollten einer genaueren Betrachtung unterzogen werden: OderKonnektor Oder Konnektor35

Fehler: Das Modell ist e-sensitiv. Folgende Prozessketten oder Elemente sollten einer genaueren Betrachtung unterzogen werden: OderKonnektor Oder Konnektor7

Fertig

(1) Selection list
(2) Options for sorting and filtering
(3) Results

Figure 16: Result window

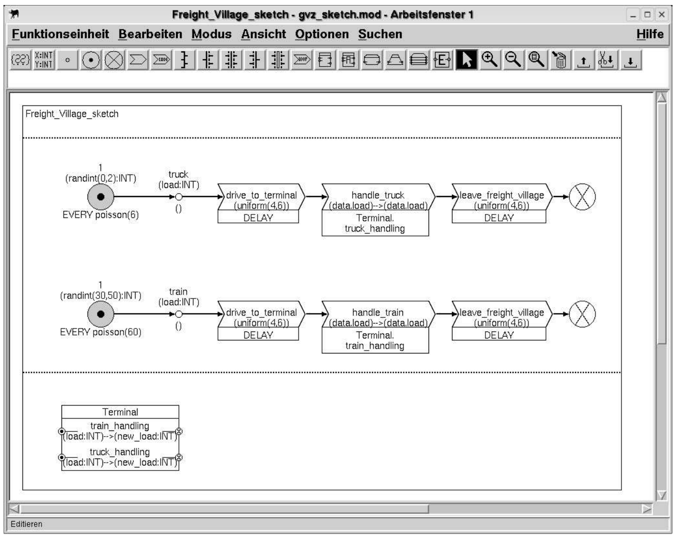


Figure 17: Results of the test for non-ergodicity for the freight village model of Sect. 2

the freight village from Sect. 2 as being non-ergodic, because the rank condition holds for the two source transitions `source_trucks` and `source_trains` (see Fig. 13). Accordingly an error-message referring to the two sources of the *ProC/B*-model will be displayed to the user, that allows an easy and fast selection of the two elements. Fig. 17 shows a screenshot from the *ProC/B*-editor with the highlighted elements, that have been identified by the test for non-ergodicity.

Of course, the model from Sect. 2 is manageable and an experienced modeller might detect non-ergodicity without performing the test. Fig. 18 shows the structural hierarchy of a larger model of a freight village. Basically the model is composed of three parts: The root of the hierarchy describes the generation of trucks and trains, that enter the freight village. The second part is a terminal for bimodal traffic, where goods are exchanged between trains and trucks. In the third part of the model mixed cargo is exchanged between trucks. A more detailed description of the model is given in [12]. As one can see in Fig. 18 there are various additional function units that model resources and subparts of the system. All in all the model reaches a level of complexity where non-ergodic situations are difficult to discover just by visual inspection. The test for non-ergodicity detects some elements (the messages are shown in Fig. 16). In this

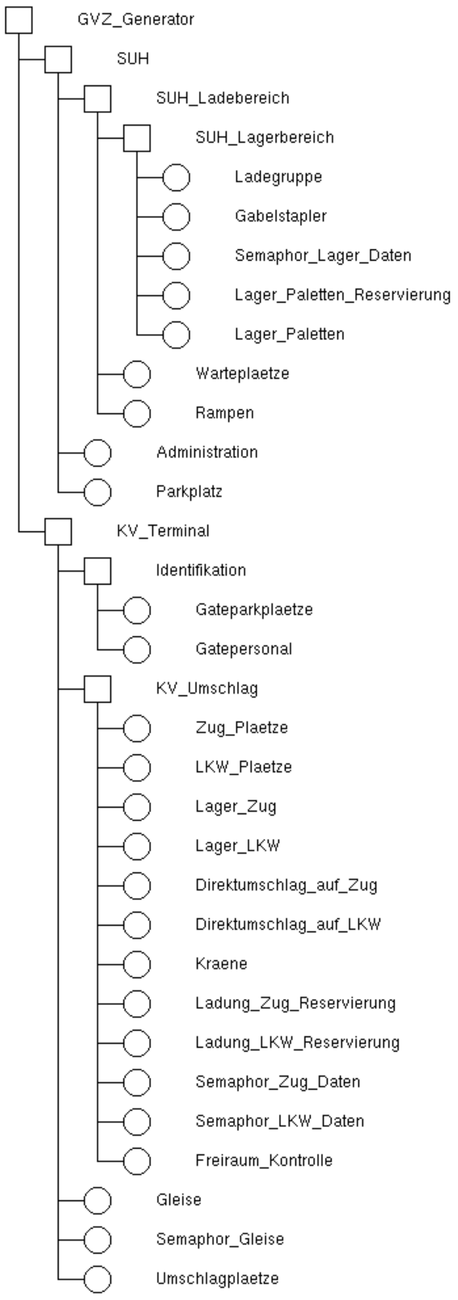


Figure 18: Hierarchy of a larger model of a freight village

case the critical PEC sets do not consist of source transitions like in the example of the small freight village model, but of transitions, that have been created by mapping connectors to Petri net elements. The situation here is similar to the setting in Fig. 6. The critical part of the model is marked grey in Fig. 15. This part of the model describes transshipping of goods between trucks and trains. The corresponding process chains are synchronised via process chain connectors and accesses to storages, which results in a sensitivity of arrival rates for trucks and trains concerning this part of the model.

It should be noted, that due to the above-mentioned limitations of the transformation from *ProC/B* to Petri nets, this might be a non-fault, but a cautious modeller can check whether the model of the freight village will behave correctly. E.g. the modeller might ensure, that trucks or trains have to leave unloaded when no synchronisation is possible, which can be modelled by conditions for the branches of OR-Connectors. Since such conditions (usually based on variables) are not strictly mapped to the Petri net, non-faults might occur. On the other hand the test for non-ergodicity will still result in a useful hint in these cases, since it points out critical parts of the model, which should be inspected with care.

6. CONCLUSIONS

We have presented typical situations in models of logistics networks which result in non-ergodic behaviour and where non-ergodicity is caused by the structure of the net and cannot be avoided by adjusting parameters. We described a method for detecting such situations that is based on a rank condition for Petri nets and can be efficiently computed. Finally we showed how this test can be transferred to the application-oriented *ProC/B* modelling world and presented the integration of the analysis technique into the *ProC/B*-toolset.

Future work will be directed towards the application of further Petri net based analysis techniques for *ProC/B*-models aiming at additional support for the validation of simulation models. Amongst functional properties like liveness, also properties in the context of restricted models might be of interest. E.g., for live and bounded extended free choice Petri nets it is known, that a marking reached after each transition has fired is a home state (cf. [8, 11]). This might help in finding the end of the transient phase or at least a reasonable starting point for sampling.

7. REFERENCES

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis. Modelling with Generalized Stochastic Petri Nets. John Wiley & Sons, 1995.

[2] F. Bause.On Non-Ergodic Infinite-State Stochastic Petri Nets. Proc. of PNPM 2003 (the 10th International Workshop on Petri Nets and Performance Tools), IEEE Society Press, Urbana (USA), pp. 84-92, 2003.

[3] F. Bause, H. Beilner. Intrinsic Problems in Simulation of Logistic Networks. Simulation in Industry, 11th European Simulation Symposium and Exhibition (ESS'99), Erlangen (Germany), October 26-28, SCS Publishing House, pp. 193-198, 1999.

[4] F. Bause, H. Beilner. A Short Note on Synchronisation in Open Systems. Petri Net Newsletter, Vol. 57, pp. 9-12, 1999.

[5] F. Bause, H. Beilner, M. Fischer, P. Kemper, M. Völker. The ProC/B Toolset for the Modelling and Analysis of Process Chains. In T. Field, P.G. Harrison, J. Bradley, and U. Harder (Hrsg.), Computer Performance Evaluation, Modelling Techniques and Tools, 12th International Conference, TOOLS 2002, London (UK), LNCS, No 2324, S. 51–70. Springer, 2002.

[6] F. Bause, P. Buchholz, C. Tepper. The ProC/B-Approach: From Informal Descriptions to Formal Models. ISoLA - 1st International Symposium on Leveraging Applications of Formal Method, 30th October - 2nd November 2004, Paphos, Cyprus.

[7] F. Bause, P. Kemper, and P. Kritzinger. Abstract Petri Net Notation. In Research Report No. 563. Fachbereich Informatik der Universität Dortmund (Germany), 1994.

[8] F. Bause, P. Kritzinger. Stochastic Petri Nets - An Introduction to the Theory. Vieweg, 2nd Edition, 2002.

[9] E. Best, K. Voss. Free Choice Systems have Home States. Acta Informatica, 21:89-100, 1984.

[10] P. Buchholz, M. Fischer, P. Kemper, and C. Tepper. New features in the APNN toolbox. In P. Kemper, editor, Tools of Aachen 2001 Int. Multiconference on Measurement, Modeling and Evaluation of Computer-Communication Systems, pp. 62–68, 2001. (Universität Dortmund, Fachbereich Informatik, Research Report No. 760.)

[11] J. Desel, J. Esparza. Free Choice Petri Nets. Cambridge University Press, 1995.

[12] C. Dilling, and M. Völker. Beispielmodellierung eines Güterverkehrszentrums im ProC/B-Paradigma. Technical Report 03016, Collaborative Research Center 559 “Modelling of Large Logistics Networks”, 2003, ISSN 1612-1376.

[13] W. Feller. An Introduction to Probability Theory and Its Applications, Volume 1. Wiley, 3rd ed., 1968.

[14] G. Florin, S. Natkin. Necessary and Sufficient Ergodicity Condition for Open Synchronized Queueing Networks. IEEE Trans. on Soft. Eng., 15(4):367-380, 1989.

[15] P. Huber, K. Jensen, and R.M. Shapiro. Hierarchies in Coloured Petri Nets. In G. Rozenberg, editor, Lecture Notes in Computer Science Vol. 483, Advances in Petri Nets 1990, pp. 313-341, Springer, 1991.

[16] F. Huber, S. Molterer, A. Rausch, B. Schatz, M. Sihling, and O. Slotosch. Tool Supported Specification and Simulation of Distributed Systems. In Proceedings of the International Symposium on Software Engineering for Parallel and Distributed Systems, 1998.

[17] K. Jensen. Basic Concepts, Analysis Methods and Practical Use, volume 1 of Coloured Petri Nets. Springer-Verlag, Berlin, 1992. ISBN 0-387-55597-8.

[18] J. Kriege. Konsistenzprüfung von ProC/B-Modellen zur Vorbereitung einer simulativen Analyse. In Proc. of the 18th Conference on Simulation and Visualization, Magdeburg, pp. 69-81, 2007.

[19] A. Kuhn. Prozessketten in der Logistik - Entwicklungstrends und Umsetzungsstrategien. Verlag Praxiswissen, Dortmund 1995.

[20] A. Kuhn. Prozesskettenmanagement - Erfolgsbeispiele aus der Praxis. Verlag Praxiswissen, Dortmund 1999.

[21] A.M. Law, W.D. Kelton. Simulation Modeling and Analysis, McGraw-Hill, 2000.

[22] J.L. Peterson. Petri Net Theory and the Modelling of Systems. Prentice-Hall, 1981.

[23] G.O. Roberts, J.S. Rosenthal, P.O. Schwartz. Convergence Properties of perturbed Markov chains. J. Appl. Prob., 35:1-11, 1998.

[24] A.W. Scheer. ARIS: Business Process Modelling. Springer, 2000.

[25] Collaborative Research Center “Modelling of Large Logistics Networks”(559). http://www.sfb559.uni-dortmund.de.

[26] E. Teruel, M. Silva. Liveness and home states in equal conflict systems. In Proceedings of the 14th International Conference on Application and Theory of Petri Nets, Chicago (USA), pp. 415-432, 1993.

[27] W. van der Aalst. Formalization and Verification of Event-driven Process Chains. Information and Software Technology, Vol. 41(10):639–650, 1999.

[28] Kees van Hee, Olivia Oanea, and Natalia Sidorova. Colored Petri Nets to Verify Extended Event-Driven Process Chains. In Proc. of the 13th International Conference on Cooperative Information Systems (CoopIS 2005), LNCS Vol. 3760, pp. 183–201, 2005.