ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 *2012. No. 7(123)*
ELEKTRONIKA IR ELEKTROTECHNIKA

SYSTEM ENGINEERING, COMPUTER TECHNOLOGY

SISTEMŲ INŽINERIJA, KOMPIUTERINĖS TECHNOLOGIJOS

Petri Nets based Production System Model Simplification

Zhijian Wang
Information Science School, Guangdong University of Business Studies, Guangzhou, P.R.China,
Guangdong Key Lab of Electronic Commerce, Guangdong University of Business Studies, Guangzhou, P.R.China, phone: 86-20-84096901, e-mail: zjian@ gdcc.edu.cn
Yuping Hu
Guangdong Key Lab of Electronic Commerce, Guangdong University of Business Studies, Guangzhou, P.R.China
Jun Zhang
Information Science School, Guangdong University of Business Studies, Guangzhou, P.R.China

crossref http://dx.doi.org/10.5755/j01.eee.123.7.2388

Introduction

Petri nets are a well-known graphical modeling tool to reflect system state transition. Petri nets can be used to describe asynchronism among different parts in the same system; at the same time more than one state transition can be fired simultaneously, so a Petri nets model is also a concurrent model. For a system described by Petri nets, its dynamic behaviors are represented by the flow of either material resources or information resources. Petri nets are regarded as one of the most important system modeling methods.

Different periods of a targeted system can be simulated using Petri nets, including system designing, constructing and executing, so as to analyze and evaluate behaviors of corresponding system. Capability and performance of a planed Flexible Manufacture System (FMS) should be evaluated during the designing process before fund investment, frequently used evaluating points include the average throughput time of parts processed in a workshop, average number of parts to be processed in the shop, the comprehensive utilization rate of equipments, the corresponding cost and profit, etc [1], above results help greatly to fund investment decision and find solution for making full use of equipment ability by combinatorial optimization.

Generalized Stochastic Petri Net (GSPN) is one of the most effective and widely used methods for problems in FMS. Performance analysis in GSPN is based on the fact that its is state space isomorphic with Markov Chain, so GSPN provides well system description and Markov Chain provides model evaluation ability based on stable mathematical foundation. We simulated an iron rolling system using hierarchical GSPN in [2], the system runs with a different length parameter each time; the ratios of failed productions to qualified productions are calculated.

The model in Fig. 1 simulates the process of part production; behaviors such as normal processing, failure and repairing are reflected in this model:

1. Places *InBuffer* and *OutBuffer* are the input and output buffer separately;
2. Places *Ready*, *Finished* and *Idle* represent state of ready to process, processing finished and machine idleness;
3. Place *Failed* represents occurrence of a mistake and the machine is in failure state;
4. Transition *Process* represents the processing process of products with the rate of λ_P and failing at the rate of λ_F;
5. Transition *Repair* represents the repairing process of machine with the rate of λ_R;
6. Transitions *Prepare* and *Release* represent the action of system preparing to produce, finishing producing and releasing resources allocated separately.

When a FMS is modeled by common Petri nets, the number of places and transitions of the model is quite large if the targeted system is relatively large. For n asynchronous process, if every one has m states, then the system has m^n states. So the state space of a complex system is enormous, using Petri nets for large scale and complex system directly will certainly cause the problem of state explosion. State explosion is the main obstacle for model analysis and verification, model complexity hinders

the application of Petri nets as an engineering method in complex systems greatly.

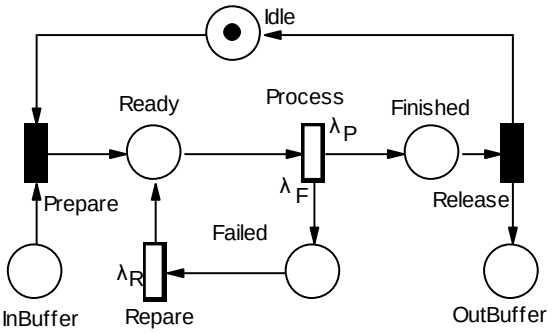

Fig. 1. GSPN model of a processing cell

To solve this problem, people presented different kinds of high level Petri nets such as Object Oriented Petri Nets (OOPN) and Colored Petri Nets [3, 4]. Usually the efficiency of above extended Petri nets are under that of the GSPNs in model calculation [2, 5], so hierarchical Petri nets are adapted in many researches and applications [5, 6]. To realize hierarchical design, abstraction and decomposition are the main methods. Abstraction is to simplify models, many states can be mapped into one so as to reduce state space and simplify model analysis. Abstraction and decomposition operate reversely; the abstraction process is from bottom to top, while the decomposition process is from top to bottom.

J. Padberg divided a complex system model into many subnets satisfying certain requirements in [7], in [8] a kind of modular Petri nets with single input and single output, which is called Controllable Output net, is defined, these Petri nets are shown to have the properties of liveness, consistency and reversibility, and the integration of such Petri nets preserves these properties under fairly weak conditions. In [9] and [10] we research the problem of how to "looks similar" between subnet and transition, namely to make their interfaces to outside environment be expressed accordantly; and the problem of "perform similarly", namely the transition and corresponding subnet should have almost the same actions.

In the rest of this paper we discuss the transformation of GSPN in order to reduce model's state space. It should be clear that although the abstracted system reserves many properties of the detailed system, it will certainly lose some of them. Fire of transition is finished for once under a state, but corresponding subnet is fired step by step, so a kind of half fired states may occur in the subnet (the detailed system). A half fired state refer to the occasion that the input transition of a subnet has fired already but the output transition does not fire yet; so some states exist in the detailed system that the tokens in the subnet's input place has been consumed but no token has been produced in the subnet's output places yet, such states have in fact no mapped states in the abstracted system, not to say properties related with such states. So it is impossible to realize absolute equity between subnet and its abstracted transition, this induce the inequality between the detailed system and the abstracted system. As for the question of under what precondition and in what degree the two systems are relatively equal, we have presented some criteria in [9] and [10].

Cell model simplification

A typical manufacturing system is composed by recoverable workstation in Fig. 1, buffers between these cells and other devices, by establishing their models we can analyze production efficiency, resource utilization and reliability of the system. We will first transform the GSPN model of a manufacturing cell in Fig. 1 into a single transition model, on this basis we can simplify GSPN models of serial processing systems and parallel systems composed by *n* machines separately.

We often need to evaluate the reliability of entire system in engineering, such as failure rate, *MTTF* (Mean Time to Failure), *MTTR* (Mean Time to Restoration or Mean Time to Repair). Let's name the machine in Fig. 1 as M, suppose its processing time (T_P), time to failure(T_F), time to repair(T_R) all depend on negative exponent distribution, its average processing rate is λ_P, average failing rate is λ_F, average repairing rate is λ_R, then T_P, T_F and T_R meet the following distribution function where λ is λ_P, λ_F and λ_R separately

$$F(t)=\begin{cases}1-e^{-\lambda t}, & t>0,\\ 0, & t\le 0,\end{cases} \qquad \lambda>0. \tag{1}$$

The Mean Time To Process (*MTTP*) is $\frac{1}{\lambda_P}$. *MTTR* is a basic measure of the maintainability of repairable items, it represents the average time required to repair a failed component or device. Expressed mathematically, it is the total corrective maintenance time divided by the total number of corrective maintenance actions during a given period of time, so $MTTR=\frac{1}{\lambda_R}$. A shorter *MTTR* is better than a longer one because its operational availability is higher.

MTTF and *MTBF* (mean time between failures) is a little different. *MTBF* is the predicted elapsed time between inherent failures of a system during operation. *MTBF* can be calculated as the average time between failures of a system. Usually people use *MTTF* to measures average time between failures with the modeling assumption that the failed system is not repaired, so $MTTF=\frac{1}{\lambda_F}$. For repairable systems, failures are considered to be those out of design conditions which place the system out of service and into a state for repair, failures can be maintained in an unrepaired condition, and do not place the system out of service any longer, in addition, units that are taken down for routine scheduled maintenance or inventory control are not considered within the definition of failure. In this paper *MTTF* is used for the repairable system since *MTTR* is considered separately, so

$$MTBF = MTTF + MTTR = \frac{1}{\lambda_F}+\frac{1}{\lambda_R}. \tag{2}$$

In this model and the rest of this paper, except the occasions with special explanations, we assume:

1. k_i , the capacity of *InBuffer* is large enough. It means when the capacity is k_i , the preceding working procedure can always transfer materials needed by following working procedure in time so that no shut-down because of material shortage would occur;
2. k_o , the capacity of *OutBuffer* is large enough. It means when the capacity is k_o , the following working procedure can always take away productions produced by preceding working procedure in time so that no block would take place because of capacity limitation;
3. Only transitions *Process* and *Repair* are timed transitions, the rest are instant transitions, that means the time of machine are spent for production process and maintenance.

Although many unimportant details have been omitted in the manufacturing cell model in Fig. 1, and the model has been simplified as much as we can do, the model still comprises 6 places and 4 transitions and looks complicated. The manufacturing cell is just the basal component of a complex system, to reduce the complexity of the cell model is the foundation to avoid the state explosion for the complete system model.

States of the model are divided into 2 types. The first type is normal states that the system runs normally, the average time to stay in normal state is *MTTF*, which is decided by λ_F ; the second type is failure state, the average time to stay in failure state is *MTTR*, which is decided by λ_R . The ratio of above two times is the system availability or generalized reliability. Availability of systems with stable state does not change with time *t*, it named as stable state availability or just availability, remarks as *A*. so

$$A = \frac{MTTF}{MTBF} = \frac{\frac{1}{\lambda_F}}{\frac{1}{\lambda_F} + \frac{1}{\lambda_R}} = \frac{\lambda_R}{\lambda_F + \lambda_R}. \qquad (3)$$

When machine failures are taken into account, the average processing time is

$$MTTP' = \frac{MTTP}{A} = \frac{\frac{1}{\lambda_P}}{\frac{\lambda_R}{\lambda_F + \lambda_R}} = \frac{\lambda_F + \lambda_R}{\lambda_P \lambda_R}. \qquad (4)$$

So the actual average processing rate with possible failures (stable state rate) is

$$\lambda_P' = \frac{1}{MTTP'} = \frac{\lambda_P \lambda_R}{\lambda_F + \lambda_R}. \qquad (5)$$

So the GSPN model in Fig. 1 can be transformed into the GSPN model in Fig. 2. The rate of transition *Process* is that in express (5), $\lambda = \lambda_P'$. If $\lambda_F = 0$ in express (5), namely the system will not fail, then $\lambda = \lambda_P' = \lambda_P$.

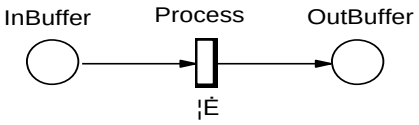

Fig. 2. The simplified GSPN model of the manufacturing cell in Fig. 1

Example. Suppose in Fig. 1, $\lambda_P = 0.1$, $\lambda_F = 0.01$ and $\lambda_R = 0.2$, then $MTTF = \frac{1}{\lambda_F} = 100$,

$MTTR = \frac{1}{\lambda_R} = 5$, $MTBF = MTTF + MTTR = 105$,

$A = \frac{MTTF}{MTBF} = \frac{100}{105} = 0.9524$, $MTTP = \frac{1}{\lambda_P} = 10$,

$MTTP' = \frac{MTTP}{A} = 10.5$,

so $\lambda_P' = \frac{\lambda_P \lambda_R}{\lambda_F + \lambda_R} = \frac{0.1 \times 0.2}{0.01 + 0.2} = 0.09524$.

Serial system simplification

The GSPN model of a serial processing system composed by *n* machines $M_i (1 \le i \le n)$ is shown in Fig. 3. *InBuffer* is the first (*0th*) buffer (input buffer for the system), *OutBuffer* is the *nth* buffer (output buffer for the system).

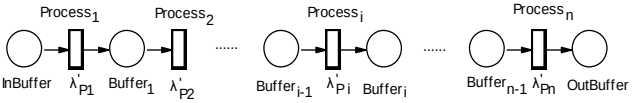

Fig. 3. The GSPN model of the serial processing system composed by *n* machines

Usually the assumption (1) and (2) in last section cannot be satisfied for a machine M_i in the serial system, material shortage and block occurs sometimes. A machine works normally when following conditions are satisfied (such states are called buffer available states, the ratio of buffer availability is represented by B_i):

1. There are tokens in the input buffer of M_i (the *(i-1)th* buffer has parts to be processed);
2. The output buffer of M_i (the *ith* buffer) is not full.

Let $\rho_i = \frac{\lambda_{Pi}}{\lambda_{Pi+1}}$, capacity of the *i*th buffer is K_i ,then according to expression(3) in [11], we know:

1. The probability that there are tokens in the *(i-1)th* buffer is

$$P_0(i-1) = \rho_{i-1}(1 - \rho_{i-1}^{K_{i-1}}) / (1 - \rho_{i-1}^{K_{i-1}+1}). \qquad (6)$$

2. The probability that the *ith* buffer is not full is

$$P_k(i) = (1 - \rho_i^{K_i}) / (1 - \rho_i^{K_i+1}). \qquad (7)$$

So the ratio of buffer availability is

$$B_i = P_0(i-1) \times P_k(i). \qquad (8)$$

Because the system only works in buffer available states, it means the machine processes or fails only in such states, so the real failure rate $\lambda_{Fi}^{'} = B_i \times \lambda_{Fi}$, substitute that in (3), then the availability of M_i is

$$A_i^{'} = \frac{\lambda_{Ri}}{B_i \lambda_{Fi} + \lambda_{Ri}}. \quad (9)$$

Use $\lambda_{Pi}^{'} = B_i \times \lambda_{Pi}$ to substitute that in (5), then the stable processing rate of M_i is

$$\lambda_{Pi}^{''} = \frac{B_i \lambda_{Pi} \lambda_{Ri}}{B_i \lambda_{Fi} + \lambda_{Ri}} = \lambda_{Pi} \times A_i^{'} \times B_i. \quad (10)$$

The stable processing rate obtained in (10) is in accordance with (11) in [11], but we get this in two steps, the first step is simplifying the manufacturing cell model, the second is calculating the buffer availability.

For a serial processing system, if the stable processing rates of different machines M_i are different, it will cause the unbalances of production ability, the machine M_k with the lowest rate forms bottleneck of the system, the processing rate is decided by $\lambda_{Pk}^{'}$, so the processing rate of the whole system is

$$\lambda_P^{'} = Min\{\lambda_{P1}^{'}, \lambda_{P2}^{'}, \lambda_{P3}^{'}, ..., \lambda_{Pn}^{'}\}. \quad (11)$$

So the GSPN model in Fig. 3 can be transformed into the GSPN model in Fig. 2, where the rate of transition Process is $\lambda_P^{'}$ in (11).

The result of (11) is different from (1) in [12], because in Fig. 3 many parts are processed at the same time, but in Fig. 1 of [12], only one part is processed each time. The complete model of Fig. 1 in [12] is shown in Fig. 4. The processing time is $t = \sum_{i=1}^{n} \frac{1}{\lambda_{P(i)}}$, which is the time sum of the *n* processes, and thus the rate is $\lambda = \frac{1}{t}$.

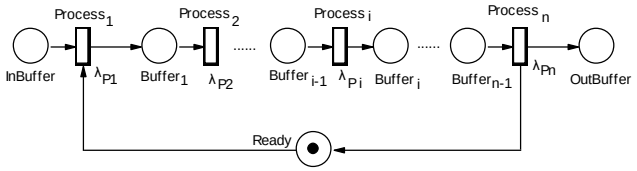

Fig. 4. Complete model of Fig. 1 [12]

A special occasion is that all buffers are with the capacity of *1*, namely *k=1*, for an arbitrary machine, $P_0 = \frac{1}{1+\rho}$, $P_1 = \rho^1 P_0 = \frac{\rho}{1+\rho}$, and $P_{\bar{0}} = P_1$, $P_{\bar{1}} = P_0$, so

$$B_i = P_{\bar{0}(i-1)} \times P_{\bar{k}(i)} = P_{1(i-1)} \times P_{0(i)} = \frac{\rho_{(i-1)}}{(1+\rho_{(i-1)})(1+\rho_{(i)})}. \quad (12)$$

If all machines own same stable processing rate, namely $\lambda_{P1}^{'} = \lambda_{P2}^{'} = .. \lambda_{Pn}^{'}$, then $\rho_{1=}\rho_{2=}..\rho_{n-1}$, according to the deducing process of (2) in [11], the possibility of the state of out buffer of M_i with *j* tokens is $P_{ij} = \frac{1}{k+1}$, so:

$$P_{\bar{0}(i-1)} = 1 - P_{0(i-1)} = \frac{k_{i-1}}{k_{i-1}+1}, \quad (13)$$

$$P_{\bar{k}(i)} = 1 - P_{k(i)} = \frac{k_i}{k_i+1}, \quad (14)$$

$$B_i = \frac{k_{i-1} k_i}{(k_{i-1}+1)(k_i+1)}. \quad (15)$$

If all machines own same stable processing rate, and all buffers are with the capacity of *1*, then according to (12) or (15), $B_i = \frac{1}{4}$ can be obtained.

If $k_{i-1} = k_i$, then some value of B_i are shown in Fig. 5. Usually large capacity buffers do benefits to improve efficiency, for example when $k_{i-1} = k_i = 9$, then $B_i = 0.81$. Of course according to (10) and (11), if processing rates of different machines are different, different buffer capacities can be assigned.

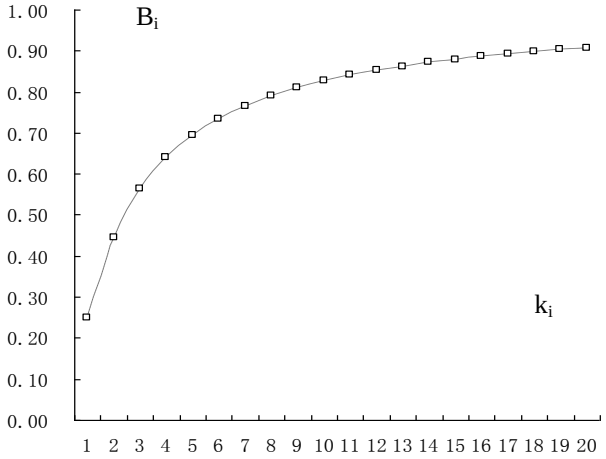

Fig. 5. Value of B_i when $k_{i-1} = k_i$

Parallel System Simplification

The GSPN model of parallel system composed by *n* machines $M_i (1 \le i \le n)$ is shown in Fig. 6.

The GSPN model in Fig. 6 can be transformed to the model in Fig. 2 where λ, the processing rate of transition *Process* is the $\lambda_P^{''}$ in (16)

$$\lambda_P^{''} = \sum_{i=1}^{n} \lambda_{Pi}^{'}. \quad (16)$$

The result of (16) is also different from (4) in [12], because in Fig. 6 parts are processed simultaneously by different machines, but in Fig. 3 of [12], only one part and one machine are selected each time, the complete model of which is shown as Fig. 7.

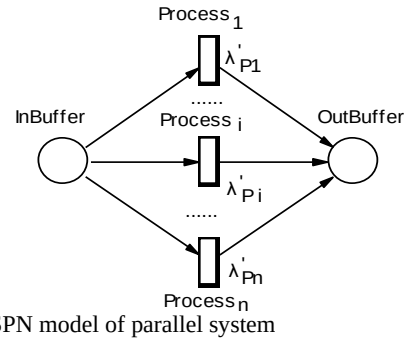

Fig. 6. The GSPN model of parallel system

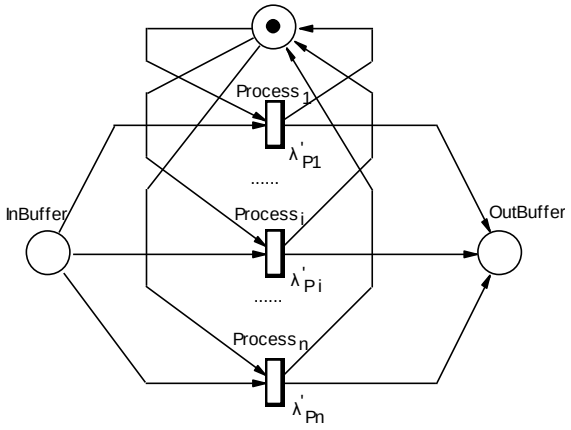

Fig. 7. The complete model of Fig. 3 in [12]

The parallel processing of Fig. 8 is in accordance with (3) in [12], according to that expression, the rate of transformed transition is the λ in (17)

$$\frac{1}{\lambda}=\sum_{i=1}^{n}\frac{1}{\lambda_i}-\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{\lambda_i+\lambda_j}+\sum_{i=1}^{n-2}\sum_{j=i+1}^{n-1}\sum_{k=j+1}^{n}\frac{1}{\lambda_i+\lambda_j+\lambda_k}+\ldots+(-1)^{n-1}\frac{1}{\sum_{i}^{n}\lambda_i} \quad (17)$$

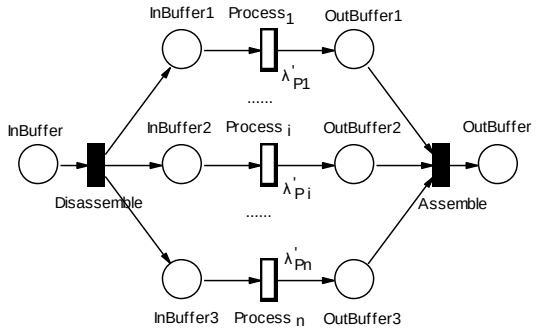

Fig. 8. A different parallel processing model

Conclusions

Petri nets is one of the most important methods used in enterprise manufacturing system, it's currently used in many related field including system simulation, performance analysis, production schedule and process control. To solve the problem of state explosion, we discussed the method to reduce model scale so that the state space can be shrunk effectively. All models, including the complete model of any system, can be transformed into the model in Fig. 2 where only one transition is used.

What should be stressed here is, although the simplified model can reserve some of the properties in the detailed system, it is not completely equivalent to the original one, some of the properties will be lost in the process of transformation. So model simplification should be used correctly, and the sub model to be transformed should be decided carefully. We have discussed this in some of our previous researches, but when some factors such as time are taken into account, this question becomes more complex, it is necessary to research it extensively in future researches.

On the investigation of a great deal of enterprise, Advanced Manufacturing Research(AMR) found those sophisticated enterprise production management systems are usually composed by three kinds of software: enterprise management software, production process monitoring software, and Manufacturing Execution System, so AMR presented a three layer enterprise integration model, the 3 layers are planning, execution and control. Although current application of Petri nets in enterprise systems covers the whole three layers, but people paid more attention to the lower layer applications in their past researches. If the problem of model complexity can be solved root and branch, Petri nets seems to be a hopeful tool which can cover all layers' functions and the complete process of enterprise manufacturing system development.

Acknowledgements

The research was supported by Natural Science Foundation of Guangdong Province (S2011010001546, S2011010001581, 10151032001000003) and Innovative Team Project of Guangdong University of Business Studies.

References

1. **Rodríguez D., Zimmermann A., Silva M.** Two Heuristics for the Improvement of a Two–Phase Optimization Method for Manufacturing Systems // 2004 IEEE International Conference on Systems, Man and Cybernetics. – No. 2. –P. 1686–1692.
2. **Wang Z, J., Cai Z. X.** A Petri Net Model for System Evaluating and Optimizing // High Technology Letters, 2003. – No. 13(5). – P. 50–54.
3. **Toshiyuki M.** A Survey of Object–Oriented Petri Nets and Analysis Methods// IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 2005. – No. E88–A(11). – P. 2964~2971.
4. **Jensen K., Kristensen L. M., Wells L.** Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems // International Journal on Software Tools for Technology Transfer (STTT), 2007. – No. 9 (3–4). – P. 213–254.
5. **Goli A. M. B., Zileh Z. H. S.** Application of Generalized Stochastic Petri Nets (GSPN) in Modeling and Evaluating a Resource Sharing Flexible Manufacturing System // World Academy of Science, Engineering and Technology, 2009. –

No. 57. – P. 356–365.
6. **Mascheroni M.** Hypernets: A Class of Hierarchical Petri Nets (Tesi di dottorato). – Università degli Studi di Milano–Bicocca, 2011.
7. **Padberg J.** Petri Net Modules // Journal of Integrated Design and Process Science, 2002. – No. 6 (4). – P. 105–120.
8. **Proth J. M.** A Class of Petri Nets for Manufacturing System Integration// IEEE Transactions on Robotics and Automation, 1997. – No. 13 (3). – P. 317–326.
9. **Wang Z. J., Wei D. G.** Modeling Complex System Using T–subnet based Hierarchical Petri Nets // Journal of Computers, 2009. – No. 4(9). – P. 829–836.
10. **Wang Z. J., Wei D. G., Xu C.** Model Transform Based on a Kind of Transition Subnet // Journal of Networks, 2010.–No. 5(6). – P. 716–723.
11. **Shu S. G.** An Analysis of the Repairable Computer Integrated Manufacturing System with Buffers and a Study of the System Reliability // Acta Automatica Sinica, 1992. – No. 18(1). – P. 15–18.
12. **Lin, C., Qu, Y.Z., Tian, L.Q.** An Approach to Performance Equivalent Simplification and Analysis of Stochastic Petri Nets // Acta Electronica Sinica, 2002. – No. 30(11). – P. 1620–1623.

Received 2012 03 19
Accepted after revision 2012 05 12

Zhijian Wang, Yuping Hu, Jun Zhang. Petri Nets based Production System Model Simplification // Electronics and Electrical Engineering. – Kaunas: Technologija, 2012. – No. 7(123). – P. 113–118.

Petri nets own graphical description ability based on solid mathematical foundation, this makes it a possible tool for analyzing complex system such as manufacturing system, but using Petri nets directly in large scale system will lead to state explosion. A Generalized Stochastic Petri Nets model of a repairable manufacturing cell is setup. As the first step to solve system state explosion problem, this 4 transitions model is transformed into a model with only one transition. Serial processing systems and parallel systems composed by such cells are transformed on the basis of cell model transformation, and such transformations are compared with those in other researches. The method presented in this paper helps to simplify system model and decrease system state space, and thus makes it feasible to use Petri nets in complex systems. Ill. 8, bibl. 12 (in English; abstracts in English and Lithuanian).

Zhijian Wang, Yuping Hu, Jun Zhang. Gamybinės sistemos modelio pagrįsto Petri tinklais supaprastinimas // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2012. – Nr. 7(123). – P. 113–118.

Petri tinklų grafinio aprašymo galimybė tvirtai pagrįsta matematiškai, todėl juos galima naudoti analizuojant kompleksines sistemas (gamybines sistemas), tačiau naudojant Petri tinklus tiesiogiai didelėms sistemoms galima prieiti iki sprogimo būsenos. Tirtas atkuriamos darbinės ląstelės apibendrintas stochastinis Petri tinklo modelis. Pradžioje sprendžiant sistemos sprogimo būsenos problemą, šis keturių perėjimų modelis transformuojamas tik į vieno perėjimo modelį. Nuoseklaus apdirbimo sistemos ir lygiagrečios sistemos, sudarytos iš tokių ląstelių, transformuotos pagal ląstelių modelio transformaciją ir tos transformacijos palygintos kituose tyrimuose aprašytomis sistemomis. Pateiktas metodas padeda supaprastinti sistemos modelį, todėl Petri tinklus galima naudoti kompleksinėms sistemoms. Il. 8, bibl. 12 (anglų kalba; santraukos anglų ir lietuvių k.).