51 A NEW METHOD FOR THE HIERARCHICAL MODELING OF PRODUCTIVE SYSTEMS

Fabrício Junqueira, Paulo E. Miyagi
Escola Politécnica, University of São Paulo, Brazil
fabri@usp.br, pemiyagi@usp.br

Industry reorganization and the increase in the automation level of productive processes results in the augmentation of complexity of the interactions among enterprise subsystems related to monitoring and control. Analysis techniques are used to deal with the complexity, design of new productive systems, and improve the performance of existing systems. In this context there is currently a special focus in distributed simulation. It deals with the execution of simulation in physically dispersed computers connected through a network. In order to explore the potential of distributed simulation, this paper proposes a new method for the hierarchical modeling of productive systems proper for distributed environment. The application of the method is illustrated through an example. This method has been successfully applied to a number of case studies in order to confirm its effectiveness.

1. INTRODUCTION

Markets are becoming global and independent of geographic barriers. More manufacture industries have been established in a distributed and dispersed way. They take advantage of the growth in the communication networks and information technology (IT), which provides the means for a larger transnational cooperation. A change in the way people considers productive systems have also been observed. New technologies and approaches provide managers and engineers with an integrated and dynamic view of productive systems. The enterprise is not considered anymore an isolated entity, but the part of a cooperative consortium of enterprises (Shi and Gregory, 1998).

The incorporation of IT in productive processes has enabled the integration of enterprise heterogeneous systems (Sanz and Alonso, 2001). As an example, an industrial supervisory system interacts with a heterogeneous set of hardware and software (workstations, remote units, programmable controllers, etc.) in order to monitor and control an industrial plant. The integration of heterogeneous systems results in an increase of system complexity.

The increase of complexity in the productive systems demands the development of new analyses solutions. Among them, the use of distributed simulation deserves particular attention. The distributed simulation deals with the execution of

simulation in geographically dispersed computers connected through a network, which results in a virtual super computer (Fujimoto, 1999; Banks et al., 2000).

In this context the purpose of this paper is to present a new modeling approach suitable for distributed simulation. This new modeling approach is based on the use of object-oriented concepts with Petri nets and progressive refinement techniques such as PFS (Production Flow Schema). The models generated by the proposed approach can be integrated and simulated concurrently with other models in a distributed and geographically dispersed environment.

Some reasons to distribute the simulation among multiple computers are (Fujimoto, 1999): (1) reduction of the execution time through the subdivision of a huge simulation model among processors; (2) integration of simulators, combining simulations that are executed in different manufacturers machines; and (3) fault tolerance, i.e., if a processor fails, other processors continue the simulation.

Kachitvichyanukul (2001) and Banks (2000) emphasize the employment of reusable models based on components. A component can be selected from a repository and used alone, or be combined with others to generate a new one.

An inherent distributed simulation problem is the partition of the model among processors. Nevison (1990) *apud* Fujimoto (1990) use previous knowledge about the modeled system to optimize the simulation. However, the optimization of the simulation becomes impracticable when working with flexible modeling tools. In this case, the modeling tool does not restrict the kind of productive systems under study. It is therefore impossible to use optimization strategies based on the previous knowledge about the system as well as the number of processors being used in the simulation.

On the other hand, models based on Discrete Event Dynamic Systems (DEDS) are intensively used to describe, analyze and control processes in productive systems. These systems are characterized by the occurrence of instantaneous events, which govern their dynamics. The evolution of state of these systems is based on rules that define the conditions for the occurrence of events as well as the new state reached after an event (Cassandras and Strickland, 1992).

Petri net is a graphical and mathematical modeling technique originally proposed by Carl Adam Petri in 1960 to characterize concurrence in computer operations (Murata, 1989; Moore and Brennan, 1996), which is a typical DEDS. Since then, it has been used for modeling dynamic systems in a large range of areas (Srinivasan and Venkatasubramanian, 1998), such as (Elkoutbi and Keller, 1998): communication protocols, distributed algorithms, computers architectures, man-machine interaction. Daum and Sargent (1999), and Kachitvichyanukul (2001), among others, stand out the hierarchical modeling as a way to deal with complex systems. The modeler can divide a complex system in subsystems (and consequently sub-models) that are better managed. Models can be generated in different abstraction levels, helping the verification and validation process.

Besides the system analysis and design considerations, the way that a productive system is modeled is based both on its inherent characteristics, such as system complexity, as well as on personal factors, such as project team experience and the intended abstraction level. In any case, the project team must be able to visualize the productive system (or the main parts under study) as a whole, its parts and behavior, and the relationship among them (their interfaces).

Based on the above consideration, in the following section, a new method for hierarchical modeling of productive system is presented. This method is proper for distributed simulation in the sense that the resulting models and its interactions are explicitly and dearly specified, exploiting the potential of distributed systems (Junqueira et al., 2005a, 2005b).

2. MODELING APPROACH

Figure 1 shows the steps of the proposed approach, which are discussed as follows.

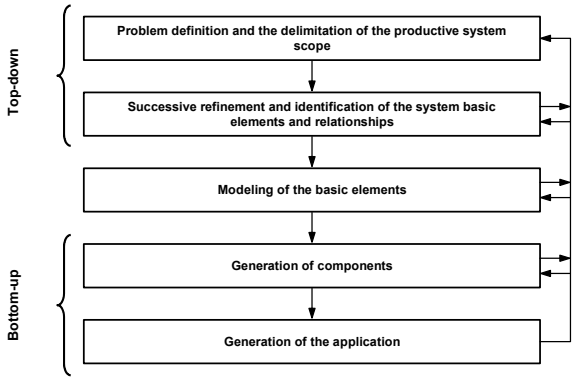


Figure 1 – Diagram of the proposed method for the modeling of productive systems.

Step 1 – Problem definition and the delimitation of the productive system scope

The modeler must delimit the scope of the productive system under study, i.e. what are the departments, equipment (tools) and people (both considered as resources) of the productive system, and what are the characteristics and processes to be modeled and analyzed.

Step 2 - Successive refinement and identification of the system basic elements and relationships

A top-down approach is adopted in this step. The use of modeling techniques such as PFS (Production Flow Schema) helps on the productive system modeling process (Hasegawa et al., 1998; Miyagi, Santos Filho, Arata, 2000). The PFS is a type of high level Petri net composed by *activities*, *distribution elements*, and *arcs*. It is a conceptual model applied in the initial phase of the system modeling process that is gradually translated into a Petri net model.

The modeling process starts at the "top" with a productive system conceptual model that is detailed until the desired level of abstraction. At the end of this step, a set of basic elements that constitutes the productive system had been identified, as well as the relationships among them, i.e., their interfaces and message exchanged formats.

Step 3 – Modeling of the basic elements

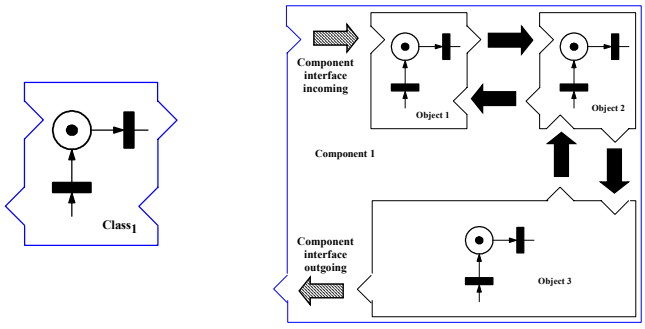
At this step, the basic elements functionalities are modeled using Petri net. Each model is called "class" (Figure 2 (a)). Similar to object-oriented program languages, a class describes a set of objects that shares the same attributes, operations, relationships, and semantics.

The model of each basic element can be analyzed isolated from the model of the remaining system, facilitating validation before its use to compose new and more complex models.

Step 4 – Generation of components

Once the basic elements (classes) have been defined, they can be combined to form more complex ones. This step may be arranged in four sub-steps: (4.1) definition of objects; (4.2) encapsulation of objects into components; (4.3) connection of object interfaces; and (4.4) mapping the remained objects interfaces as components interfaces.

The componentization process starts (4.1) using each class as a template to generate one or more objects. A bottom-up approach is adopted, and objects that share some common features, or need to work together to perform a task, are grouped (4.2) to constitute a component (Figure 2 (b)). Then, (4.3) object interfaces are connected (black arrows in Figure 2 (b)). For the proposed method, in a Petri net model, the interfaces are modeled as transitions and the relationship among models are done through transitions fusion (Figure 3) (Sibertin-Blanc, 1993).



(a) (b)

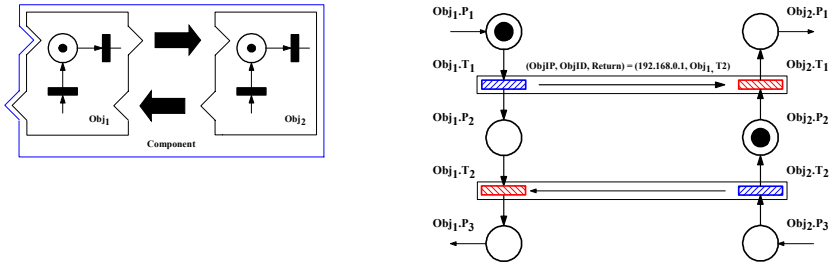
Figure 2 - (a) a class modeled as a Petri net; and (b) a component constituted by three objects.

To conclude the component model, it is necessary (4.4) to map the remaining object interfaces as the component interfaces. The downward diagonal arrows in Figure 2 (b) are examples of this mapping.

Step 5 – Generation of the application

To generate an application, two or more components are grouped and their interfaces are connected (Figure 4). This step is similar to the previous one. The difference is that applications do not have external interfaces. In other words, making analogy

with software elements, a stand-alone component does not execute anything and may be used in different contexts, while an application has all the necessary elements to work alone and has a well-defined purpose.



(a) (b)

Figure 3 – Example of object interface: (a) schematic representation; and (b) Petri net representation with transitions fusion.

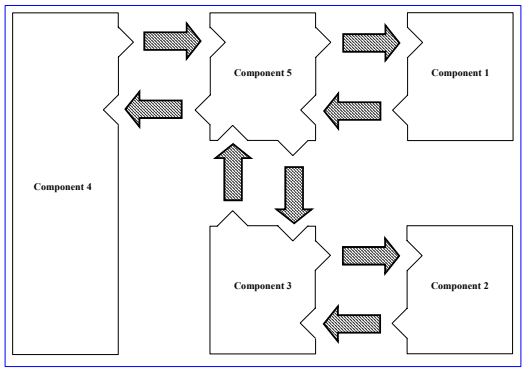


Figure 4 – An application composed by two or more components.

3. A MATERIAL TRANSPORTATION SYSTEM EXAMPLE

To illustrate the application of the method, a practical example is presented. The complete models are in Junqueira (2006); in this paper it is presented a sample of some significative aspects of the method.

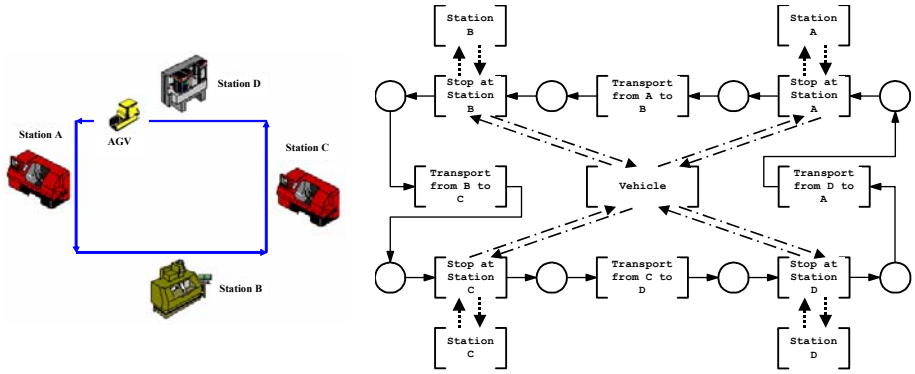
Step 1 – Problem definition and the delimitation of the productive system scope

The material transportation system is composed by four stations. Each station one produces or consumes goods. Station A produces goods for Station B, and Station C produces goods for Station D. A vehicle, with unitary capacity, is used for the transportation of goods as showed in Figure 5.

Each station has a processing time. Stations A and C delivery a good (and Stations B and D get a new good) when they finish their processing time. The mechanism to transfer goods from a vehicle to a station and vice-versa is not considered in this example. However, the transfer time is modeled as a vehicle characteristic, i.e., the vehicle will remain stopped until the conclusion of transfer.

Step 2 - Successive refinement and identification of the system basic elements and relationships

Based on the described system (Figure 5(a)), the PFS is used to model the main activities as well as to detail its relationships. Figure 5(b) is the transcription of Figure 5(a) system schema to PFS. It represents the vehicle circular path and the vehicle passage through each station. Some activities of this model may be highlighted: (1) [Transport between stations]; (2) [Station stop] (where goods are (un)loaded); (3) [Vehicle]; and (4) [Station]. Moreover, square dot arrows represent the information exchanged between [Station stop] and [Station], and the long dash arrows, the information between [Station stop] and [Vehicle].



(a) (b)

Figure 5 – (a) vehicle circular path, passing through the Stations; and (b) material transportation system PFS.

Step 3 – Modeling of the basic elements

A Petri net model of the activity [Transport between stations], where the stations are generically identified by X (departure) and Y (arrive) is presented in Figure 6. Place P1 is a pre-condition of this operation. Transition T1 considers the time needed to go from X to Y. Place P2 is the pos-condition of the operation.

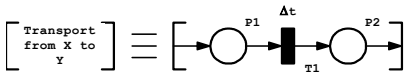


Figure 6 – Petri net model of [Transport between two stations (from X to Y)].

Figure 7 presents the stop at each station (generically called Station Z). Place P1 signs the arrival of the vehicle to the station. The operation beginning is represented by transition T1. Place P2 represents the (un)load process. T2 signs the end of the operation, and place P3 represents the finished state of the operation.

The activity [Vehicle/Station interface] is detailed in Figure 8. Place P1 represents the initial state of the (un)load operation. P2 signs the (un)load operation, and P3, its end. Transitions T1 and T2 are, respectively, the beginning and end of the (un)load operation.

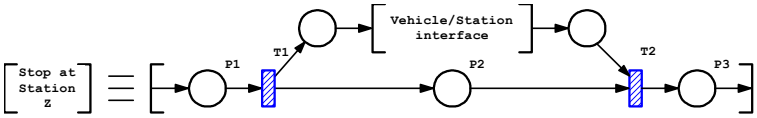


Figure 7 – Petri net model of [Stop at Station Z].

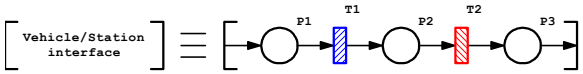


Figure 8 – Petri net model of [Vehicle/Station interface].

Figures 9 (a) presents the Petri net model of [Station Z]. Transitions T1 and T2 are the [Station Z] model interfaces. T1 represents the beginning of the (un)load operation while T2 represents its end. Transition T3 represents the (un)load operation duration. Place P1 represents the station waiting for the end of the (un)load operation. Places P3 and P2 are, respectively, the beginning and end of good processing operation.

The [Vehicle] Petri net model is showed in Figure 9 (b). Place P1 represents the traveling state of the [Vehicle], while P2 and P3 are respectively the beginning and end state of the (un)load operation. Transition T1 represents the vehicle stop and the (un)load operation beginning. T3 represents the (un)load operation duration, and T2, the (un)load operation end. T1 and T2 are also the interfaces of [Vehicle] model.

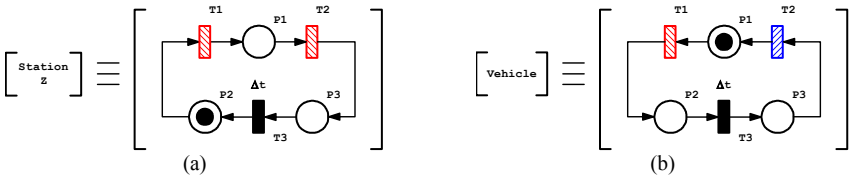


Figure 9 – (a) Petri net model of [Station Z]; and (b) Petri net model of [Vehicle].

The relationships between [Stop at station Z] and [Station Z] models are made through the fusion of transitions (Figure 10). Transitions T1 of both models are merged, working as a single one. The same is done for T2.

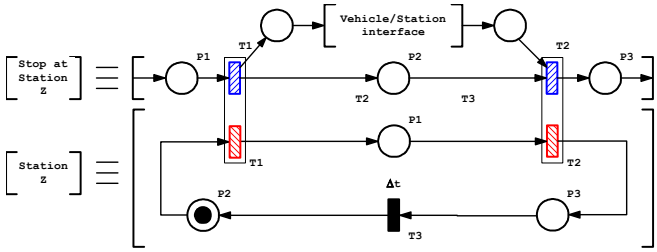


Figure 10 – The relationship between the activities [Stop at station Z] and [Station Z] through transition fusion.

Figure 11 illustrates [Vehicle] and [Vehicle/Station interface] transitions fusion. Transitions T1 of both models behave as an only one. The same occurs with their transitions T2.

Analyzing Figures 5 (b), 6, and 7, a simplification can be applied to part of the model detached on Figure 12 (a). In this example, place P2 of [Stopped in C] model and P1 of [Transport from C to D] model (Figure 12 (b)), for example, can be merged, resulting in the place P1 of the Figure 12 (c) model.

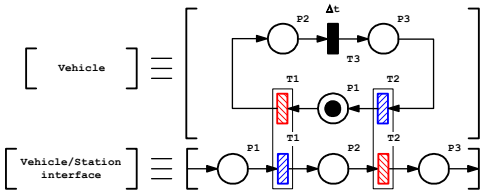


Figure 11 – The relationship between the activities [Vehicle] and [Vehicle/Station interface] through transition fusion.

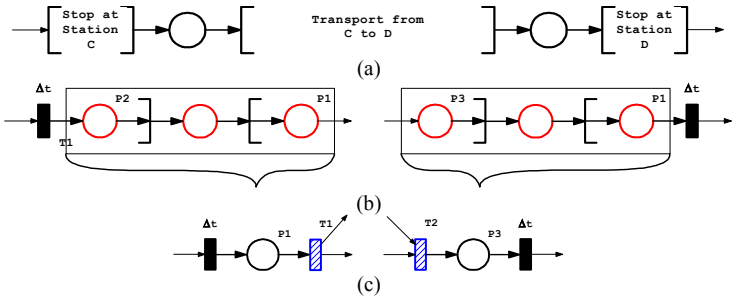


Figure 12 – Application of techniques to reduce the model.

The simplification technique is applied to the model of Figure 5 (b), which results in the [Path] model (Figure 13). Another simplification adopted in this example is the use of only one pair of transitions (T13 and T14) to interface with the [Vehicle] model. Therefore, conflicts for resources become evident in the model and the component generation (step 4) is simplified.

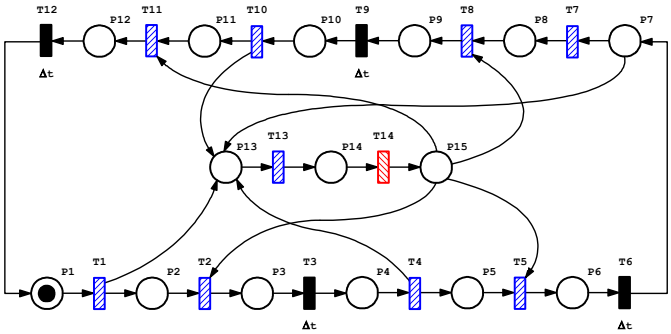


Figure 13 – Petri net functional model of Figure 5 (b).

Step 4 – Generation of components

Figures 14 (a), (b) and (c) show, respectively, the AGV objects (based on the class Vehicle), Station A (B, C e D) (based on the class Station), and Path network (based

on the class Path).

Then these objects are grouped to compose the Manufacture cell component (Figure 14 (d)). The UML component interface notation is used on Figure 14 object models to show the exchange of information among them.

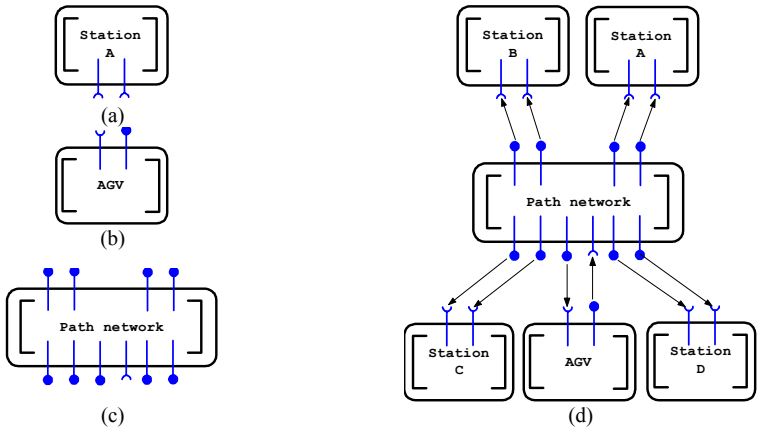


Figure 14 – (a), (b) and (c) are objects based on the classes defined at the step 3; and (d) manufacture cell component.

Step 5 – Generation of the application

For this simplified example, the application model is the same as the component one.

4. CONCLUSIONS

As observed for Daum and Sargent (1999), Kachitvichyanukul et al. (2001) and Banks (2000), researches in the modeling and simulation area are necessary, especially in the direction of developing new hierarchical modeling techniques, as well as the reuse of models.

The proposed modeling technique makes possible the progressive productive system refinement and understating, through successive steps. This approach allows a better characterization of the system elements and the relationships among them. Elements that possess common characteristics can be represented by a single model, guaranteeing its reusability. Concurrently, a library of models may be implemented. In this level, the properties and functionalities of each element can be verified. Then, from the composition of these elements, more complex elements, such as subsystems, can be created, and its functionalities, validated. The whole system model is obtained through successive compositions.

Moreover, this method is proper for distributed simulation, once resulting models and its interactions are explicitly and dearly specified. Thus, distributed simulation can be optimized since models can be better distributed among processors.

5. ACKNOWLEDGMENTS

The authors would like to thank the partial financial support of the Brazilian governmental agencies CNPq, CAPES, and FAPESP, specially the TIDIA/KyaTera committee.

6. REFERENCES

1. Banks, J. (Chair) Simulation in the future. In. Proceedings of the 2000 Winter Simulation Conference, 2000; 1568-1576.
2. Cassandras, CG, Strickland, SG. Sample Path Properties of timed Discrete Event Systems in Discrete Event Dynamic Systems – Analyzing Complexity and Performance in the Modern World. New York: IEEE Press, 1992.
3. Daum, T, Sargent, RG. Scaling, hierarchical modeling, and reuse in an object-oriented modeling and simulation system. In. Proceedings of the 1999 Winter Simulation Conference, 1999; 1470-1477.
4. Elkoutbi, M., Keller, RK. Modeling Interactive Systems with Hierarchical Colored Petri Nets. In: Proceedings of the 1998 Advanced Simulation Technologies Conference, 1998; 432-437.
5. Fujimoto, R. Parallel Discrete Event Simulation. Communications of the ACM, 1990; Vol. 33, No. 10, 30-53.
6. Fujimoto, RM. Parallel and distributed simulation. In. Proceedings of the 1999 Winter Simulation Conference, 1999; 122-131.
7. Hasegawa, K et al. Enhanced production flow schema for modeling the complex resource sharing system. In: BASYS'98 IEEE/IFIP Int. Conf. on Information Technology for Balanced Automation Systems in Manufacturing, Praha, 1998; 335-354.
8. Junqueira, F., Villani, E., Miyagi, PE. A Platform for Distributed Modeling and Simulation of Productive Systems based on Petri Nets and Object-Oriented Paradigm. In: ETFA´05 10th IEEE. International Conference on Emerging Technologies and Factory Automation, Proceedings, IEEE, Catania, Italy, 2005a; 907-914.
9. Junqueira, F., Villani, E., Miyagi, PE. A petri net based platform for distributed modeling and simulation of productive systems. In: COBEM2005 International Congress of Mechanical Engineering, Proceedings. Ouro Preto, Minas Gerais, Brazil; 2005b.
10. Junqueira, F. Modelagem e simulação distribuída de sistemas produtivos. Doctor Thesis, Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos, São Paulo, 2006.
11. Kachitvichyanukul, V. (Chair) Simulation Environment for the new millennium. In. Proceedings of the 2001 Winter Simulation Conference, 2001; 541-547.
12. Miyagi, PE, Santos Filho, DJ, Arata, WM. Design of deadlock avoidance compensators for anthropocentric production systems. In: BASYS'2000 IEEE/IFIP Int. Conf. on Information Technology for Balanced Automation Systems in Production and Transportation, Berlin, 2000; 287-294.
13. Moore, KE, Brennan, JE, Alpha/SIM Simulation Software Tutorial. In: Proceedings of the 1996 Winter Simulation Conference, 1996; 632-639.
14. Murata, T. Petri Nets - Properties, Analysis and Applications. In: Proceedings of the IEEE, 1989; Vol. 77, No. 4.
15. Nevison, C. Parallel simulation of manufacturing systems: Structural factors. In Proceedings of the SCS Multiconference on Distributed Simulation 22, 1990; Vol. 1, 17-19.
16. Sanz, R, Alonso, M. CORBA for control systems. In: Annual Reviews in Control, 2001; No. 25, 169-181.
17. Shi, Y, Gregory, M. International manufacturing networks – to develop global competitive capabilities. Journal of Operations Management, 1998; No. 16, 195-214.
18. Sibertin-Blanc, C. A Client-Server Protocol for the Composition of Petri Nets. In: Proceedings 14th International Conference on Application and Theory of Petri Nets. Chicago, Illinois, USA, 1993; 377-396.
19. Srinivasan, R, Venkatasubramanian, V. Automating HAZOP analysis of batch chemical plant: Part I. The knowledge representation framework. Computers Chem. Engineering, Great Britain, 1998; Vol. 22, No. 9, 1345–1355.