43 COMPUTATIONAL REPRESENTATION, HETEROGENEITY AND INTEGRATION OF PRODUCTION SYSTEM MODELS

Wilson M. Arata and Paulo E. Miyagi
Escola Politécnica da Universidade de São Paulo
wimarata@usp.br, pemiyagi@usp.br

This work discusses important aspects in the computational representation of models, focusing on the treatment of the heterogeneity and the integration of models. The relevance of this topic lies in the necessity of achieving an efficient workflow when handling heterogeneous information structures and of giving proper answer to the involvement of new types of models driven by the increasing demand for enhanced information handling capabilities in the planning and the control of production systems.

1. INTRODUCTION

This work presents some points to consider when planning and designing computational support systems that make intensive use of models, taking into account the heterogeneity and integration models. In the case of this text, these issues are discussed in the context of the so called Discrete Event Dynamic Systems (DEDS), from which perspective important results for production systems have been obtained. DEDS are systems whose dynamics presents discrete states and the transitions between the states are associated to the occurrence of discrete events (Cassandras, 1993). In the case of production system, the coordination of operations and the resource utilization patterns can be approached from the perspective of DEDS.

Models are an important means to approach systems, providing an effective way to deal with their complexity. They are abstract descriptions of systems, that consider only the relevant aspects (for a certain purpose), ignoring those that are irrelevant.

Models are inherently specific in the sense that their structure and content strongly depend on the aspects that are considered, the mathematical tools used to deal with them and the purposes of their utilization. This specificity is a major factor in the heterogeneity, that is, existence of a wide variety of models, expressed in terms of different modeling formalisms. In the case of DEDS, this heterogeneity is remarkable, with some examples presented in Table 1.

Table 1 – Brief descriptions of some DEDS types of models

Petri Nets (Murata, 1989)	Modeling of causal relationships between different state variables, providing information on qualitative features.
Stochastic Petri Nets (Molloy, 1980)	Extension of Petri Nets with stochastic timings associated to state transitions, to approach performance features of systems.
Markov Chains (Kulkarni, 1995)	Description of state transitions and the timings associated to them, for the study of state probabilities.
Queueing Networks (Bolch, 1998)	Analysis of congestion phenomena as entities seeks for service when accessing a network of limited resources.
Mark Flow Graph (Hasegawa, 1988)	Extension of Petri Nets considering discrete production system control elements.

The idea behind computationally supporting model heterogeneity is to take advantage of the best that each type of model ca provide and, thus, enhance the planning and the operation of production systems. It is also important to remember that the diversity of modeling techniques is continuously growing. Therefore, it is important to develop an adequate computational framework that does not amplify the complexity inherent to the model heterogeneity, providing a favorable cost-benefit ratio.

Besides the existence of heterogeneous models, there is the issue of integration of models, that is based on the consideration of the relationships between different models. A well-known example is the case of the isomorphism between a Stochastic Petri net and a Markov chain (Molloy, 1980): if both models refers to a same system dynamics, it is very likely that the former constitutes a much more concise description, while, from the latter, that can be computationally obtained from the former, one can calculate quantitative information on the system dynamics. So, another aspect to consider is the adequate treatment of the relationships between models, in order to maximize the advantages that those relationships can provide.

The focus of this work is on the issue of the computational representation of the information conveyed by the models and how it can put into effect the benefits of an adequate treatment of heterogeneity and integration. So, at first, a structured discussion about the characteristics a modeling language should have to deal with model heterogeneity is carried out; for that propose, set-based diagrams are presented as a means to visualize relevant aspects in modeling and analysis and the relationships between them. In the second part, the representation of semantic information associated to models is presented as a means of dealing with the integration of models, showing that, if adequately elaborated, it can provide a unified view of different models referring to a same system dynamics, where the diverse relationships between models are shown without ambiguity, redundancy and concealment of relevant information. Besides analytical models, these ideas can be extended to other information structures, such as those containing sensorial data, so that it can be treated together with other models within a computational framework.

2. HETEROGENEITY AND WORKFLOW EFFICIENCY

This section introduces an abstract model of modeling and analysis tools to serve as an instrument to achieve a structured discussion about the characteristics of computational representations of models and of their handling. It uses basic elements of Set Theory and provides a useful way to visualize important relationships, being general enough to approach a wide range of configurations of computational modeling and analysis frameworks or schemes.

So, consider that all computational modeling and analysis related activities take place within Modeling and Analysis Environments (MAEs), whose constitution is described in detail in the next subsection. Also, models are computationally handled in the form of numerical-symbolic constructs. Ultimately, the role of a MAE is to process numerical-symbolic constructs and generate numerical-symbolic constructs. The numerical-symbolic constructs are given the generic denomination of *structures* in the text. In particular, the structures containing all the information provided by a model are called *representations*.

2.1 Mathematical description of MAEs and mapping models into them

Mathematically, a MAE can be described by a triple $\langle L_C, L_B, C_A \rangle$, whose elements, explained in the following paragraphs, can be represented by diagrams, like those present in Figure 1.

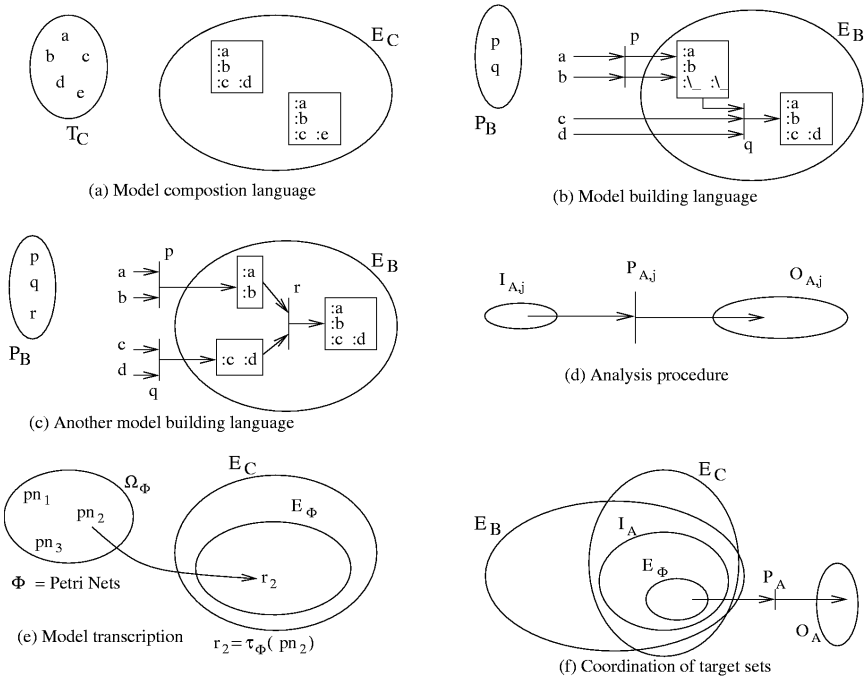

Figure 1. Diagrams involving target sets

$L_C=\langle T_C,E_C\rangle$ is a *model composition language* that provides the descriptive elements that comprise model representations, where T_C is the set of terms from which structures are made and E_C is the set of all valid structures (according to a certain criterion) — Figure 1(a) shows examples of T_C and E_C, the latter being a set of model representations expressed, in this example, by forms whose fields are correctly filled with the elements from the former.

$L_B=\langle P_B,E_B\rangle$ is a *model building language*, such that P_B is the set of operations on T_C (in L_C) provided by this language and E_B is the set of all structures that can be built using operations in P_B and terms in T_C — Figure 1(b) shows examples of P_B, E_B and the application of some operations in P_B in the construction of filled and semi-filled forms that belong to E_B; in Figure 1(c), another example shows a building language that provides operations to construct fragments of forms (operations *p* and *q*) and an operation *r* to compose complete forms from fragments.

$C_A = \{C_{A,1}, ..., C_{A,n}\}$is a set representing the *analytical capacity* of the MAE, where $C_{A,j}$ is a triple $\langle P_{A,j} ,I_{A,j} ,O_{A,j}\rangle$, where $P_{A,j}$ is a *procedure* implementing an analysis, $I_{A,j}$ and $O_{A,j}$ are, respectively, the sets from which inputs are taken and to which the results from the procedure belong — Figure 1(d) schematically illustrates $C_{A,j}$.

In this text, mentions to modeling language usually refer to the pair $\langle L_C,L_B\rangle$.

Along with mathematical models of MAEs, types of models are introduced by means of a *transcription function* $\tau_{\Phi,:}\ \Omega_\Phi \rightarrow E_C$, where Φ stands for a model type (like Petri Nets or Markov Chains), Ω_Φ for the set of all models of type Φ and E_C for the set of all valid models in L_C. E_Φ is the image of τ_Φ , i.e. the set of all representations of models of type Φ in the language L_C.

Some of the sets just introduced are called *target sets* and they are specially interesting for the purposes of this work, for representing four essential dimensions in the relationships between MAEs and types of models:

- E_C is the target set of the composition language, representing the model expressivity of the MAE;
- E_B is the target set of the model building language, representing the model building features of the MAE;
- $I_{A,j}$ is the target set of procedure $P_{A,j}$, referring to the analyzability of structures as provided by the implementation of analysis methods;
- E_Φ is the target set of the transcription of models of type Φ, that is, indicating how models of this type are represented within a MAE.

An effective MAE must have its target sets duely coordinated, so that, from the point of view of the representations of the models of a type Φ to be considered, they can be expressed, built and analysed. The configuration illustrated in Figure 1(f) meets these conditions: $E_\Phi \subset$ in E_C means that the MAE can effectively express such models of type Φ; $E_\Phi \subset E_B$, that it can build representations of these models; $E_\Phi \subset I_A$, they can computationally processed by the analysis implemented by the MAE.

An interesting feature that can be observed in certain model building languages (L_B) is that, if appropriate operations are provided in P_B, the construction of model representations can be performed with the use of several operations. In this case, besides the aimed model representation, the process generates intermediary structures, as illustrated in Figures 1(b) and 1(c). The availability of such structures

grants flexibility to the model building process, since more building paths can be followed and diverse structure reuse pattern can be employed.

When dealing with heterogeneous models, Figure 2 shows two ways of handling them. In Figure 2(a), each type of model is handled by a specific MAE, as indicated by the target sets. In Figure 2(b), the configuration that is the object of interest of this work is shown, where all types of models are represented using a comprehensive modeling language. A modeling language is said to be *comprehensive* if it can express and build models of multiple types (it is interesting to notice that Figure 2(b) also includes the results of the procedures). The problem with the specific MAEs configuration is that, in the general case, implementation and learning costs with respect to model composition and building languages can be significant and the transition from working with a specific MAE to another can be cumbersome. The configuration described in the next paragraph can minimize these problems.

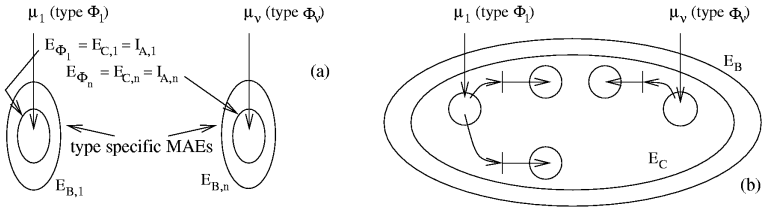

Figure 2. Configurations for dealing with heterogeneous models: (a) specific MAEs and (b) comprehensive MAE.

Thus, going further, a modeling language is ideally *comprehensive and uniform* if, with the same set of elements and composition rules, is capable of building representations of models of an indefinite number of types. This is particularly important when a new type of model is to be considered: a uniform MAE (Figure 3(a)) incorporates new model types without changes; a nonuniform MAE (Figure 3(b)) probably will need extensions in the expressive part, which will likely induce a need for extensions in the building language as well.

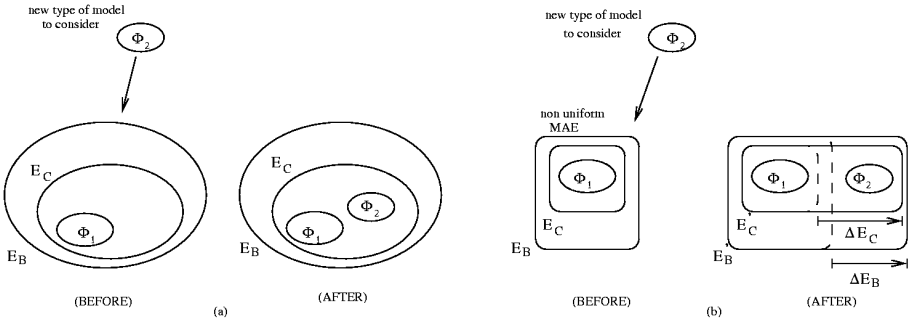

Figure 3. Comprehensive MAEs: (a) uniform and (b) nonuniform.

Then, at this point, it is possible to enumerate important features that an extensible MAE for heterogeneous models should present: coordination between the elements of MAE and the transcriptions of different types, comprehensive and uniform model composition language, flexible model building language and analytical extensibility (to incorporate new analysis procedures). Also, there are issues like representation ambiguity and redundancy, that are treated in (Arata and Miyagi, 2003).

3. INTEGRATION OF MODELS AND THEIR SEMANTICS

Comprehensive and uniform modeling languages are important to avoid the amplification, by the computational modeling and analysis frameworks, of the complexity due to model heterogeneity. However, even languages presenting such features cannot prevent the inherent formal heterogeneity of models. This heterogeneity means that, if no support is available, dealing with heterogeneous models means having to deal with information organized in different structures, what can lead to many very different procedures to access and manipulate the contents of each of them.

In the direction of such support, it is interesting to consider the fact that the integration of models (specially those of different types) is based on the semantic bindings between them (Arata and Miyagi, 2005). In this context, semantics refers to the meaning, that is, to what is being represented by the models; therefore, the semantics of models corresponds to what is observed in the system and in the dynamics being treated. Indeed, the simple fact that different models refer to one same dynamics suggests the existence of such connections.

A notorious case of model integration involves the isomorphism between Generalized Stochastic Petri Nets (GSPNs) and Continuous-time Markov Chains (CTMCs) (Marsan, 1984): if a GSPN is isomorphic to a certain CTMC, there is an one-to-one mapping *f* between GSPN markings and CTMC states so that transitions from markings *M* to *M'* mean that there are transitions from CTMC states $S=f(M)$ and $S'=f(M')$. In formal terms, GSPN marking is a concept strange to CTMCs in the same way a CTMC state is with respect to GSPNs; so, the nature of this binding is essentially semantic, that is, the only fact that links related GSPN markings and CTMC states is that they refer to the same entities, i.e., the same dynamic states.

Based on the fact that semantic bindings are a major component behind model integration, it is natural to expect that a proper representation of the semantic information associated to the integrating models plays a role in the development of a description that reflects and makes explicit the integration of models. In the next section, this representation is elaborated in the form of a set of predicates and it is shown that, while formal comprehensiveness and uniformity of modeling languages provides an effective coexistence of heterogeneous models, a coordinated representation of semantic information provides a straightforward way to present the relationships between different models, overcoming a barrier due to heterogeneity.

A major issue in building such descriptions is dealing with semantic conflicts. In particular, two kinds of name conflicts can make the descriptions ineffective: synonyms and homonyms.

In the case of synonyms (multiple names, one entity), the description may not completely reflect the integration, since certain relationships can be interpreted as involving different entities, while, instead, they involve one same entity (for example, if an entity A is denoted by both X and Y, the expressions "X is being processed" and "Y is being inspected" give no clue about the fact that both situations refers to entity A; allowing synonyms requires additional elements to handle their occurrences, increasing the complexity of the descriptions).

If homonyms (one name, multiple entities) are allowed, semantically ambiguous or senseless statements can be made (if X refers to a machine and also to a part, an order to "mill X" can lead to undesired situations).

The consideration of semantic bindings is an important element in the representation of integration of models. Although not explicitly mentioned, this is also the mechanism used by the hierarchical model composition in the SHARPE system (Trivedi, 2002), where results of model analysis are employed as elements of other models, and in the Möbius framework (Sanders et al., 2003), where formalisms are described in terms of the components provided by the framework.

4. APPLICATION OF THE CONCEPTS

In this section, the concepts thus far develop are illustrated by means of the example of the analysis of GSPNs via isomorphism with CTMCs (Marsan, 1984), similar to the case described in the previous section. Briefly describing, in this analysis, a GSPN model is used to enumerate the relevant conditions to the dynamics being modeled, how these conditions enable state transitions and what conditions these transitions maintain, activate or deactivate; it also specifies the duration of the activation of the conditions. Then, a timed reachability graph is built, describing what conditions are active in each reachable state, indicating the elements associate to each state transition. From this graph, a CTMC model is generated and, then, its steady-state probability distribution is calculated (providing the steady-state probability of each reachable state).

A comprehensive and uniform modeling language is presented in (Arata and Miyagi, 2003). The language provides three kinds of constructs that are used to construct all the data types that can be expressed by the language: atoms, homogeneous sets (whose elements are all of the same type) and tuples (ordered aggregates of objects), where the latter two can be nested one inside another indefinitely. In Figure 4, using these constructs, metamodels for GSPN, timed reachability graphs, CTMC and steady-state probability distributions are defined.

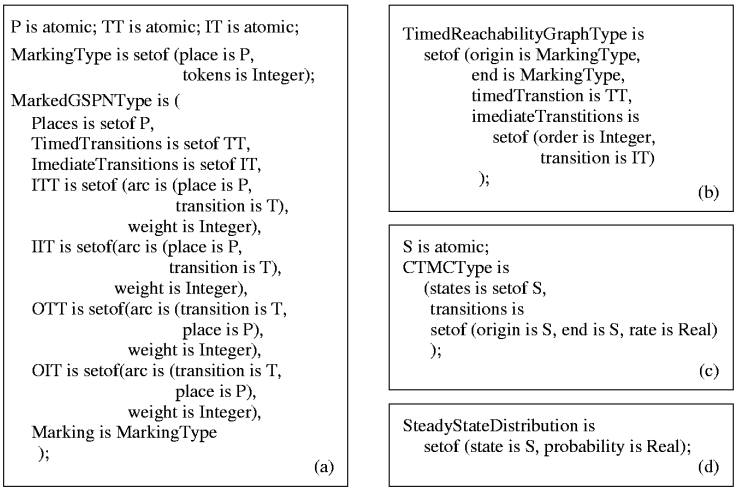

Figure 4. Metamodels describing representations of models

Examining the first line of the metamodel in Figure 4(a), three atomic types (referring to atomic objects or atoms) are defined: P, TT and IT; the idea is that other data types are built using atoms of these three type. The type MarkingType is defined to be a set (using the keyword **setof**) of elements, whose type follows the keyword **setof**: in this case, the elements are tuples (that are specified by a list of type specification enclosed by parentheses); these tuples are of a data type with two elements: the first one, that can be referred by the name place, is of type P, and the second one, that can be referred by the name tokens, is of type Integer (a number, that the language treats as a special kind of atom). The same can be analogously said about the other metamodels. So, this is a modeling language that can express a wide variety of types of models (comprehensiveness), using the same simple elements and constructs (uniformity).

In Figure 5, instances of these types of models implementing such analysis are shown. Figure 6 presents a set of predicates, similar to those in the Prolog language (Deransart, 1996), that is a representation of the semantic information associated to the models in Figure 5. The "rate" predicates specify the rate of completion of an activity, described by a timed transition in the GSPN. The "in" predicates state that a certain dynamic state belongs to the state space of the system (denoted by the term "StateSpace"), what can be observed from both GSPN and CTMC. The "at" predicates indicates that a certain activity occur in a certain state, what can be concluded from the places of the GSPN and the marking in the timed reachability graph. The "prob" predicates indicates the probability of occurrence of a certain state, obtained from the steady-state probability distribution of the CTMC. Consistency in this representation is achieved by using the same terms for states and the same expressions for activities in the different predicates obtained from different models

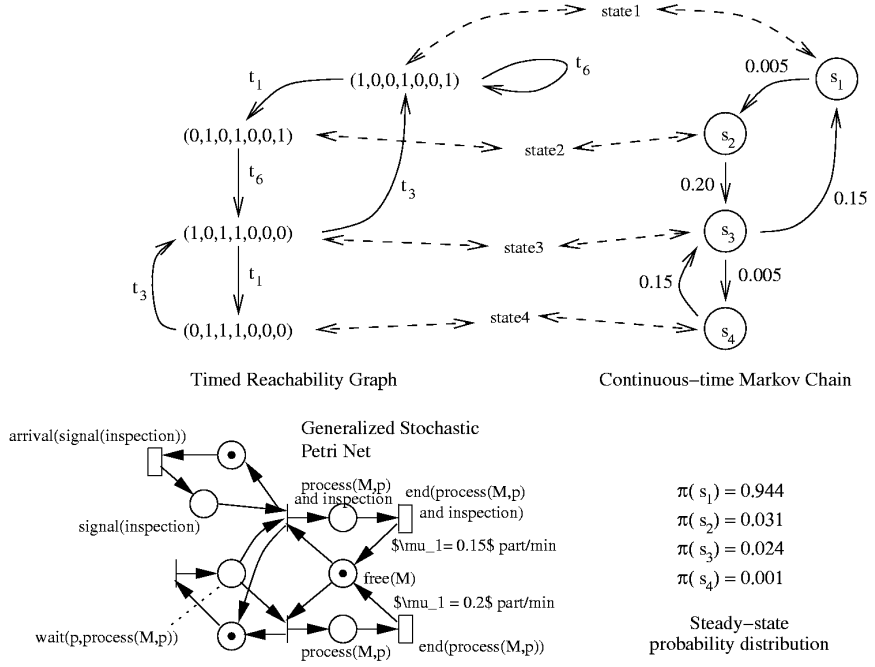

Figure 5. Models involved in the analysis of GSPN via isomorphism

```
rate(process(M,p),.2).
rate(process(M,p) and nspecting,.15).
rate(arrival(signal(inspection)),.005).

in(state1,StateSpace).
in(state2,StateSpace).
in(state3,StateSpace).
in(state4,StateSpace).
```

```
at(process(M,p),state1).
at(process(M,p) and inspecting,state2).
at(process(M,p) and inspecting,state3).
at(process(M,p),state4).

prob(state1,0.944).
prob(state2,0.031).
prob(state3,0.024).
prob(state4,0.001).
```

Figure 6. Representation of semantic information via predicates.

The use of predicates in the representation of semantic information shows that information carried from different models can be put in one same description, in a way that the heterogeneity of models is no longer an obstacle to visualize the relationships between the entities that participate in the system dynamics, as long as consistency in the representation is observed.

5. CONCLUSIONS

This work presents some guidelines to be observed in the design of computational systems dealing with heterogeneous models. Adequate treatment of heterogeneous models is relevant as it enables an efficient usage of computational and human resources by providing an infrastructure for a smoother workflow when

dealing with model heterogeneity. It also covers the case where new models need to be considered (for instance, because new features are to be included in a system); thus, supporting heterogeneity increases the likelyhood of a computational tool to evolve following a smooth path. Diagrams involving target sets have shown to be a useful means to visualize features such as comprehensiveness and uniformity. Also it has been shown that an adequate representation of semantic information associated to model integration leverages the contribution that multiple models working together can give, providing more resources to query and manipulate those information.

6. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the financial support to the present project provided by the Brazilian Governmental Agencies CNPq, CAPES and FAPESP. Particularly, the authors would like to thank TIDIA/KyaTera program under which this work is developed.

7. REFERENCES

1. Arata WM, Miyagi PE. Uniform computational treatment of heterogeneous discrete-event dynamic system models. Proceedings of 9th IEEE International Conference on Emerging Technologies and Factory Automation, Lisbon, Portugal, 2003; p.47-53.
2. Arata WM, Miyagi PE. Formal comprehensiveness and uniformity and semantic intra and intermodel consistency in the representation of Discrete Event Dynamic System models. Proceedings of the 18th International Congress of Mechanical Engineering, COBEM 2005, Ouro Preto, Brazil, 2005.
3. Bolch G, Greiner S, Meer H, Trivedi KS. Queuing Networks and Markov Chains: modeling and performance evaluation with computer science applications. New York: Wiley-Interscience, 1998.
4. Cassandras CG. Discrete Event Systems: Modeling and Performance Analysis. Burr Ridge: Richard D. Irwin Inc,1993
5. Deransart P, Cervoni L, Ed-Dbali A. Prolog: the standard: reference manual. London: Springer-Verlag, 1996.
6. Hasegawa K, Takahashi K, Miyagi PE. Application of the Mark Flow Graph to represent discrete event production systems and system control. Transactions of the Society of Instrument and Control Engineers 1988: 1, 69-75.
7. Kulkarni VG. Modeling and Analysis of Stochastic Systems. London: Chapman & Hall, 1995.
8. Marsan MA, Conte G, Balbo G. A class of generalized Stochastic Petri Nets for the performance evaluation of multiprocessor systems. ACM Transactions on Computer Systems 1984: 2, 93-122.
9. Molloy MK. Performance analysis using Stochastic Petri Nets. IEEE Transactions on Computers 1980: 9, 913-917.
10. Murata T. Petri Nets: Properties, Analysis and Applications. Proceedings of IEEE 1989: 4, 541-580.
11. Sanders WH, Courtney T, Deavours D, Daly D, Derisavi S, Lam. Multi-formalism and Multi-solution-method Modeling Frameworks: The Möbius Approach. Proceedings of the Symposium on Performance Evaluation - Stories and Perspectives, Vienna, Austria, 2003: 241-256.
12. Trivedi KS. SHARPE 2002: Symbolic Hierarchical Automated Reliability and Performance Evaluator, Proceedings of 2002 International Conference on Dependable Systems and Networks (DSN 2002), http://csdl.computer.org/comp/proceedings/ dsn/2002/1597/00/15970544.pdf.