UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação
ISSN 0103-2577

Stochastic Statecharts for Obtaining Performance Measurements of a File Server Model

Carlos Renato Lisboa Francês
Nandamudi Lankalapalli Vijaykumar
Marcos José Santana
Solon Venâncio de Carvalho
Regina Helena Carlucci Santana

Nº 51

NOTAS

Série Computação

São Carlos – SP
Dez./2000

Resumo. Statecharts são considerados uma técnica de especificação de alto nível, idealizadas originalmente para representar sistemas complexos. Statecharts derivaram dos tradicionais diagramas estados-transições, entretanto, com a adição de características de hierarquia e paralelismo entre os estados. Este artigo apresenta uma nova e original utilização para os Statecharts: a avaliação de desempenho. Essa nova abordagem é possível graças à associação entre um modelo Statecharts e sua cadeia de Markov equivalente, o que possibilita que, através do uso de métodos analíticos, sejam obtidas medidas de desempenho. São apresentados dois exemplos de servidores de arquivos especificados em Statecharts, além dos seus correspondentes em redes de filas e redes de Petri. Resultados numéricos das medidas de desempenho também são mostrados.

Stochastic Statecharts for obtaining performance measurements of a File Server model[1]

Carlos Renato Lisboa Francês[1], Nandamudi Lankalapalli Vijaykumar[2], Marcos José Santana[3], Solon Venâncio de Carvalho[2], Regina Helena Carlucci Santana[3]

[1]Department of Computer Science, CCI, University of Amazônia, CEP 66060-000, Belém, PA, Brazil.
frances@icmc.sc.usp.br.
[2]Laboratory of Computing and Applied Mathematics, LAC, Brazilian Institute of Space Research, INPE, C.P. 515, CEP 12201-970, São José dos Campos, SP, Brazil.
{vijay, solon}@lac.inpe.br.
[3]Department of Computer Science, ICMC, University of São Paulo, C.P. 668, CEP 13560-970, São Carlos, SP, Brazil.
{mjs, rcs}@icmc.sc.usp.br.

Abstract. Statecharts are considered as a high-level specification tool to represent complex systems. It descends from the state-transition diagrams but enhances the representation by incorporating notions of hierarchy and parallelism. The objective of this paper is to show that this tool may also be used to represent performance models and by generating the equivalent Markov chain performance measurements are obtained by making use of analytical numerical methods. Examples of two models of File servers represented in queuing networks and Petri nets will also be specified using Statecharts tool. Numerical results of performance measurements will be shown.

1 Introduction

In many a situation, it is interesting to know a priori information about the behavior of a given system to be developed. It is also a fact that systems, already developed, are also a target to be studied so that evaluation of their performance is determined. Modeling has become an important topic of research as the representation of a given system allows an insight of how the system behaves. As examples, distributed systems, manufacturing systems and many others may be mentioned.

The scope of this work concentrates on systems that are known as complex as they are reactive. Reactive systems are those that the behavior of the system changes at a given instant according to a perturbation of the system. These types of systems are everywhere in the daily routine such as telephones, cars, communication networks, operating systems, etc.

---

[1] Paper published in the International Conference on Information Society in the 21 Century: Emerging Technologies and New Challenges, University of Aizu, November 5 - 8, 2000, Aizu-Wakamatsu City, Japan.

In order to evaluate the performance of a given system, solutions may be obtained through two approaches: The first one is simulation and the other category is analytical methods. The scope of this paper is concentrated on solutions via analytical methods. Usually, the analytical methods are based on Markov theory. It is essential to denote that the main concern of this paper is the modeling of a given system in a clear manner and at the same time it should be possible to be treated computationally.

A very first natural solution of representing a reactive system is by selecting state-transition diagrams. If the events among the states follow an exponential distribution, then this structure may be considered as a Markov chain. Once a system is represented by state-transitions diagrams and by applying appropriate numerical methods [9] and [10], one can determine the steady-state probabilities with which performance measurements may be obtained. However, the use of state-transition diagrams may be cumbersome especially when dealing with parallel components as they may lead to exponential blow-up of the model. This problem leads to the necessity of coming up with better representation tools.

There are already widely used tools to cope up with the representation of rather complex systems and these are queuing networks [7] and Petri nets [8]. Originally, queuing networks have been developed to observe two fundamental factors: requests that are queued for a given service and a resource that provides the service. Petri nets are also very popular and a very great deal of work has been done using this method. Some extensions to Petri nets, Stochastic Petri nets and Generalized Stochastic Petri nets, are available to deal with performance models.

The paper here explores the use of a new approach in adopting Statecharts [3], [4], [5] and [6] to represent performance models. The question is how to solve the specified model in order to obtain the performance measurements. The solution is in generating its equivalent representation in state-transition diagrams, a Markov chain, that has a one-to-one correspondence with a matrix structure. This matrix is the input to the available numerical methods to determine the steady-state probabilities, basis to yield performance measurements [1].

The idea of adopting Statecharts to represent and treat performance models is another alternative to specify a given system in a high-level of abstraction. This is achieved due to the basic features provided by Statecharts that are extremely useful in depicting most of the necessities (hierarchy, parallelism, entry by history, etc.) essential in representing performance models. In order to show this capability, a software system was developed [13] and [12]. This system took into consideration those elements provided by Statecharts formalism that are essential to represent a Performance Model. Other elements of the formalism not considered for the developed software system are part of an ongoing research project to enhance it.

The next section describes in a very brief manner the features of Statecharts. The section III shows the method used to treat the represented system to obtain the performance measurements. Then, section IV is devoted to show two examples of a file server [11] represented in queuing networks, and Petri nets. In both these representations, some performance measurements are obtained for the two example models and are shown. The same examples are also represented in Statecharts and by making use of analytical approach performance measurements are obtained and compared with those obtained from queuing networks and Petri nets.

2 Statecharts

Complex systems consist of subsystems working in parallel. In order to model these systems ideas of parallelism, resource sharing, synchronization, interdependence, hierarchy and randomness (random perturbations) are to be considered. Statecharts are graphics-oriented and capable of specifying such reactive systems. This tool is an extension of state-transition diagrams by adding concepts of hierarchy (depth), orthogonality (representation of parallel activities) and interdependence (broadcast-communication), i.e., it is an attempt to address both depth and modularity problem as well as the concurrency problem [3] and [4]. The basic elements that make part of this high-level specification tool are: States, Events, Conditions, Actions, Expressions, Variables, Labels and Transitions. Among these basic elements, it is necessary to give a brief explanation of the interpretation of Events as they play an important role when generating the Markov chain.

Event is a very important element to observe the model of a given system. It is considered as an interference to the system in the sense that the present system behavior as a whole is changed to another behavior with its (event's) occurrence. Statecharts provide some special events such as true (condition), false (condition), entered(X) and exited(X) (these special events are abbreviated in the Statecharts formalism as tr(condition), fs(condition), en(X) and ex(X) respectively) to cope up with the internal logic of the modeled system. The first two are respectively true and false if the condition is evaluated to be true and false.

The last two are related to a state X in which the event en(X) is stimulated if state X is entered whereas the event ex(X) is stimulated whenever there is an exit from state X. These four special events are considered as immediate events which means that these are the first to be reacted automatically in any given system.

Statecharts formalism classifies events into two categories: external and internal events. In the case of performance models, it is decided to keep these categories with the following definitions. External events are the stochastic events (where time between their activation and their occurrences follow a stochastic distribution) that have to be externally stimulated to yield new configurations. Internal events are those special (immediate) events mentioned above (true (condition), false (condition), entered(X) and exit(X)) where they are always checked and continuously reacted until none of them are active. Actions are also considered as internal events. This discussion will be detailed in the next chapter.

All the resources and formalism of this powerful tool Statecharts can be seen in [3] and [4].

3 Construction of a Continuous-Time Markov Chain from a Statecharts Model

A Markov Chain – more precisely, within the scope of this work, a Continuous-Time Markov Chain - consisting of transition rates among states is the input to the available numerical methods [9] and [10] to determine the steady-state probabilities.

It has been already mentioned that a one-to-one correspondence can be made between a Continuous-Time Markov Chain and a state-transition diagram. Therefore,

the problem of constructing the Markov model will be solved if the model represented in the high-level specification tool Statecharts generates the state-transition diagram that corresponds to the specified model. As can be seen, with the state-transition diagram, it is possible to produce the transition matrix from which steady-state probabilities are determined and these probabilities are the basis to calculate the performance measurements.

Once the model is specified in a Statecharts representation, the enabled events must be stimulated so that new configurations are yielded. Internal events are generated by Statecharts semantics and considered as immediate events in the sense they are immediately stimulated when the model reaches a configuration where they are active.

Events can also be triggered by actions associated to transitions and they (actions) are stimulated as soon as the transition is fired. External events are events that are not generated by the Statecharts semantics and therefore they must be stimulated for describing the dynamics of the modeled system behavior. In order to make the association of Statecharts model with a Markov chain possible, the only type of external events that can be considered are stochastic events. These events are those that the time between their activation and their occurrence follows a stochastic distribution. In particular, for Continuous-Time Markov Chains, this distribution has to be exponential distribution.

Once the model is specified in Statecharts representation, the first step is to take the initial configuration provided by entries by default. In other words all the initial States of each orthogonal component are taken. A reaction, that is stimulation of events to yield a new configuration, is first done by stimulating the so called immediate events such as true(condition) and false(condition). This reaction results in a new configuration. If there are actions associated with these immediate events, these are fired yielding new configuration. This process of reacting to immediate events continuously takes place until a configuration is obtained and it is such that no more immediate events are active. Now, a list of stochastic events is obtained based on the resulted configuration. New configuration is generated reacting to each stochastic event of the list. Actions are executed if they are associated with these stochastic events. This new configuration is checked against any active immediate events. This process goes on until all the configurations have been expanded. The results of all this process is a list of a structure that contains a source configuration, stochastic event (along with its rate) stimulated and the target configuration. With this information a transition matrix has to be generated with which steady-state probabilities can be determined.

These configurations have to be organized in such a way that a transition matrix consisting of transition rates can be generated, i.e., the rows and columns of the matrix are configurations and the elements correspond to transition rates from a source configuration to a destination configuration. This will be the input for the existing library to calculate the steady-state probability vector and consequently the performance measurements of interest.

In order to understand the complete functioning of the process from the specification until the generation of the transition matrix an example will be shown. Consider a system with three parallel components that correspond to two machinery equipment and a supervisor to repair any failure in the equipment. The components E1 and E2 denote the equipment whereas the component Supervisor is responsible for repairing the equipment when they fail and a priority is provided to repair E1

whenever both the equipment are down. The list of stochastic events include a1, r1, f1, s1 a2, r2, f2 and s2. Internal events are tr[in(B1)], tr[in(B2) ^ ¬in(B1)]. Actions c1 and c2 that are fired once the events s1 and s2 are taken are also considered as internal events. This model is shown in Figure 1.

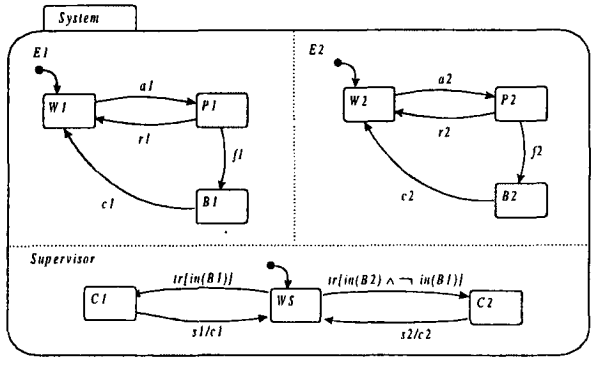

Fig. 1. Statecharts representation of equipment with a repairer.

Once the specification is over, a first reaction is performed by first checking the internal events that may be enabled according to the initial configuration and then stimulating these enabled events. For example, internal events such as tr[in(X)], where X is a State, are reacted if the system's initial configuration has an active State X. In the case of the example presented in Figure 1 no internal events are active for the initial configuration. Therefore, taking the initial configuration (W1, W2, WS) the enabled events are stochastic events a1 and a2. Therefore, these events will be stimulated to yield the new configurations.

When the system is in the initial configuration and a1 is activated, the next configuration is (P1, W2, WS). Suppose that during the process of stimulating the events, a configuration is (B1, B2, WS). In this case the active events are the so called immediate events provided by tr[in(B1)] and tr[(in(B2) ^ ¬in(B1)].

As informed earlier, these are the events that have to be checked and enabled before the stochastic events, i.e. even though there are enabled stochastic events, the immediate events (true(condition) and false(condition)) are the events that have to be stimulated.

The state-transition diagram is shown in Figure 2. As one can notice, this diagram consists of only stochastic events that follow exponential distribution. The states and arcs with events following exponential distribution compose a Markov chain with which numerical methods can be applied to determine steady-state probabilities, the basis for calculating performance measurements.

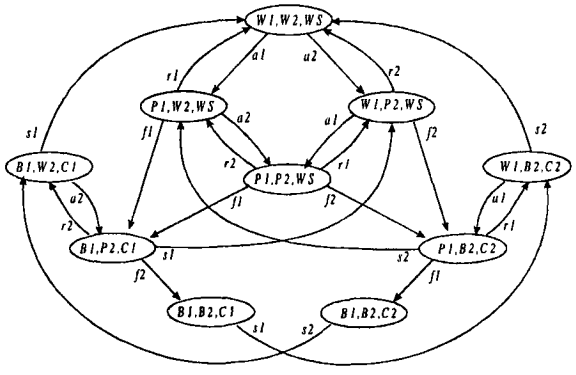

Fig. 2. State-transition diagram of Figure 1.

As the above figure is a Markov model, a one-to-one correspondence with a transition matrix can be associated. By organizing the rates among the configurations obtained in the figure, a transition matrix is formed. This transition matrix, consisting of the transition rates, is the input to the library of classes and by invoking the appropriate methods the steady-state probabilities are determined.

4 Examples of a File Server

This section is devoted to specify two examples of a File server using queuing networks, Petri nets and Statecharts. Results of performance measurements obtained by these three representations are also shown for both of these models.

As a first example, consider a file server that consists of a processor. Whenever the processor is busy, a request to use the processor is put in a processor queue. After a request has been processed, it may leave the system with a probability of 0.6 or it may use the disc with a probability of 0.4. The disc also makes use of a queue, disc queue, for storing the requests whenever the disc is in use. After the request finishes occupying the disc service, it is returned to the processor. Figures 3, 4and 5 show the specification of the File System in queuing networks, Petri nets and Statecharts respectively.

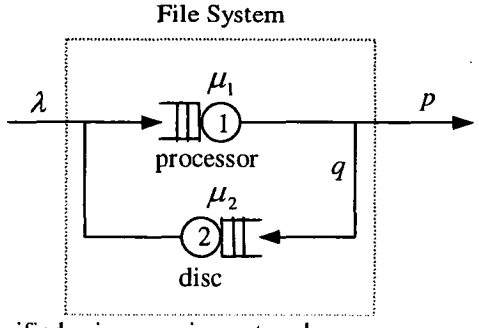

Fig. 3. File server specified using queuing networks.

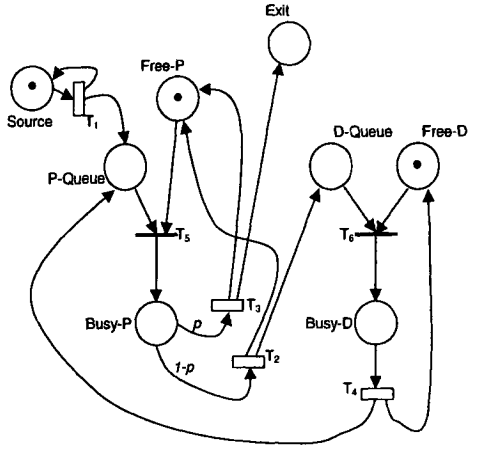

Fig. 4. File server specified using Petri nets.

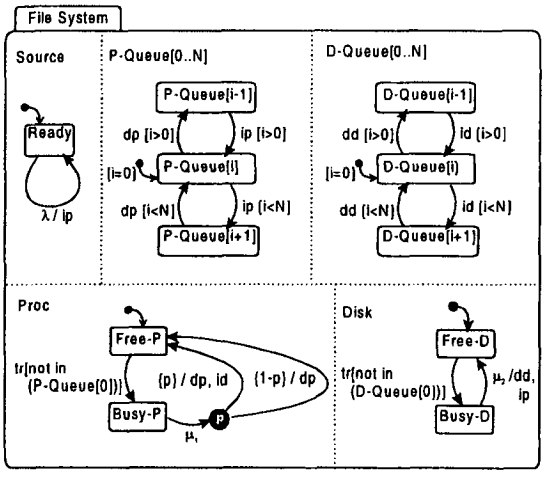

Fig. 5. File server specified using Statecharts.

The input data used to produce the performance measurements are shown in Table 1.

Table 1. INPUT DATA FOR THE SPECIFIED SYSTEM

Arrival rate λ	12
Processing rate μ1	25
Disc rate μ2	20
Probability of leaving the system p	0.6
Probability of using the Disc q = 1-p	0.4

The results of performance measurements obtained for each specification are shown in Table 2.

Table 2. PERFORMANCE MEASUREMENTS

States	Queuing nets	Petri nets	Statecharts
Proc. Busy	0.800	0.799	0.800
Proc. Idle	0.200	0.200	0.199
Disc Busy	0.400	0.401	0.400
Disc Idle	0.600	0.598	0.599

The results of performance measurements using queuing networks were based on Jackson networks whereas Petri net solution were derived from simulation. In case of Statecharts, the software system developed uses the analytical approach to determine the performance measurements. The buffer limits to store the requests in both the Processor and Disc queues were considered unlimited in the case of queuing networks and Petri nets. However, this is not possible yet using Statecharts. Therefore, the results obtained using Statecharts specification in Table 2 used the buffer limits as 40 for the Processor queue and 20 for the Disc queue.

By changing the buffer limits other interesting measurements were obtained for Statecharts specification. These results are shown in Table 3.

Table 3. PERFORMANCE MEASUREMENTS WITH BUFFER LIMITS FOR STATECHARTS

Proc. Queue	**5**	**10**	**20**	**30**
Disc Queue	**3**	**5**	**10**	**15**
Proc. Busy	0.728	0.781	0.798	0.800
Disc Busy	0.384	0.397	0.400	0.400
Proc. Idle	0.271	0.218	0.201	0.199
Disc Idle	0.615	0.602	0.599	0.599

The Table 3 shows that the Processor remains busy by increasing the buffer size for storing the processor requests in the Processor queue. However, this is not true. The results shown in Table 2 are based on the buffer size limits of 40 and 20 for Processor and Disc queues respectively. The gain in the Processor usage is not very

significant even after increasing the size of the buffer, i.e., once the buffer size for the Processor queue reaches 30, it is not advantageous to increase its size as it will not increase the usage of the Processor.

As a second example, two elements are added to the same model presented in the first example. These additional elements refer to Input Front-End Processor (PFE-In) and Output Front-End Processor (PFE-Out). The PFE-In is responsible for communication management between the network and the file server. Each incoming request to the model is first received by PFE-In. On the other hand, PFE-Out looks after the communication management between the file server and the network.

This example, specified in queuing networks is presented in Figure 6. The circle with number 1 –center 1 - within it represents PFE-In whereas PFE-Out is indicated as center 4. Centers 2 and 3 represent the processor and the disc respectively. Just as in the case of the first example, there is a probability p of the request to proceed to PFE-Out and a probability q of needing an access to the Disc.

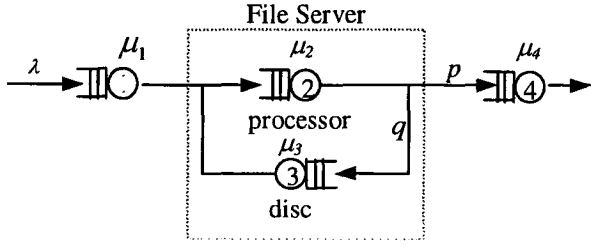

Fig. 6. File server with Input and Output Front-End processors in queuing networks.

The input values for each center and probabilities used for this model are shown in Table 4.

Table 4. INPUT DATA FOR THE SPECIFIED SYSTEM

Arrival rate λ	12
Processing rate in PFE-In μ1	25
Main processor's processing rate μ2	25
Disc rate μ3	20
Processing rate in PFE-Out μ4	18
Probability of proceeding to PFE-Out p	0.6
Probability of Disc access q = 1-p	0.4

This same example is also shown represented in Petri nets and Statecharts respectively in Figures 7 and 8.

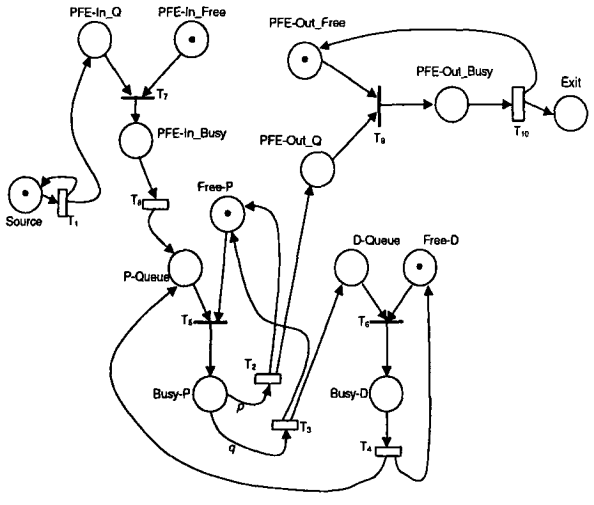

Fig. 7. File server with Input and Output Front-End processors in GSPN.

The results of performance measurements obtained from the representations of queuing networks, Petri nets and Statecharts are shown in Table 5. Here also analytical solution of Jackson network was applied to queuing network specification whereas a simulation using the DNAnet tool [2] was used to produce the results for the Petri net specification. In the case of Statecharts, analytical approach was taken to produce the performance measurements.

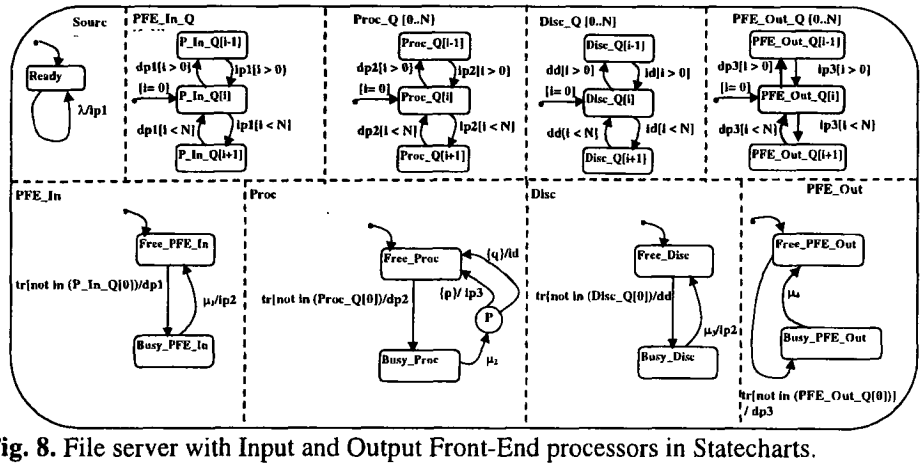

Fig. 8. File server with Input and Output Front-End processors in Statecharts.

The results of performance measurements obtained from the representations of queuing networks, Petri nets and Statecharts are shown in Table 5. Here also analytical solution of Jackson network was applied to queuing network specification whereas a simulation using the DNAnet tool [2] was used to produce the results for the Petri net specification. In the case of Statecharts, analytical approach was taken to produce the performance measurements.

As can be seen from the Table 5, there are differences in the obtained performance measurements using Statecharts specification. The main reason is the limitation of buffers. The library to solve the Markov chain restricts the space state to be finite. Therefore, the buffers used in the example showed in Figure 8 must have a limit. The number of buffers used here was 5 for the PFE_In Queue, 10 for the Processor Queue, 5 for both Disc Queue and PFE_Out Queue. In the model specified by Statecharts, some jobs are lost in case the buffer used for PFE_In Queue is full. One more interesting fact to mention is the use of the processors (Front-End and the main processor) as well as the Disc. The guarding condition used to leave the busy state depends on the availability of the buffer queues. This means that the busy state will be on until a space (in the buffers) is available even though the job has been processed. This leads to a more usage of the processors.

Figure 9 shows the flows of the jobs among the components of the system in which the buffers are limited in size with the values mentioned previously.

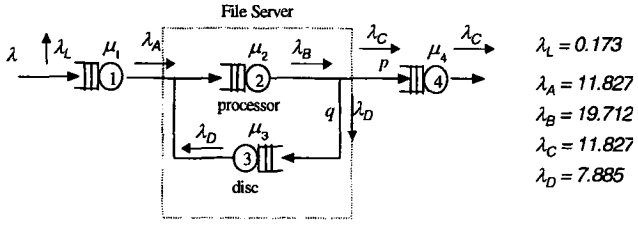

Fig. 9. Job Flow among the components of the system with limited buffer size

In order to make the model yield the same results as in queuing networks and Petri nets, it is possible to increase the size of the buffers among the components. However, price for the computational effort must be paid. This specific model yielded 4116 configurations and 25536 transitions. This graph with a large number of nodes and arcs gives an indication that use of high-level specification tools to represent a model and automatically convert it into a Markov chain (when using analytical approach) is duly justified. Efforts are being made in order to improve the library that solves a Markov chain especially regarding the restriction of a finite state space.

Table 5. PERFORMANCE MEASUREMENTS

States	Queuing nets	Petri nets	Statecharts
Processor Busy	0.800	0.801 +/- 2.721601e-05	0.821
Processor Idle	0.200	0.198 +/- 2.393877e-05	0.178
PFE_In Busy	0.480	0.477 +/- 2.193415e-05	0.513
PFE_Out Busy	0.666	0.667 +/- 2.636289e-05	0.657
Disc Busy	0.400	0.395 +/- 2.112764e-05	0.437
Disco Idle	0.600	0.604 +/- 3.181774e-05	0.563
Number of Clients in the PFE_In Queue	0.443	0.426 +/- 3.871829e-05	0.540
Number of Clients in the Processor Queue	3.2	3.051 +/- 0.000183	3.039
Number of Clients in the Disc Queue	0.2667	0.248 +/- 2.758534e-05	0.352
Number of Clients in the PFE_Out Queue	1.360	1.295 +/- 9.290842e-05	0.978
System *Throughput*	12	(T10) 12.012 +/- 0.123139	11.827

5 Comments

This paper showed another alternative of representing performance models. Specification of complex systems is not an easy task and it is not the intention of this paper to claim that Statecharts are the best option. However, the potential features provided in Statecharts open a wide range of intricacies to be specified in the model. One more interesting factor is that Statecharts are an extension of state-transition diagrams that are very closely associated to Markov chains which are very much used in analytical solutions to determine the performance measurements.

At the moment, a main program has to be written to specify a Statecharts model for the software system. A user interface is one of the top priorities to be added to the developed software. There are two interesting features, very useful in representing performance models, provided in the Statecharts formalism that are entry by History and parameterized states. The next step is to include these features in the software system. Long-run plans do exist to study the feasibility of representing Markov Decision Models.

6 Acknowledgments

The authors gratefully acknowledge the precious help of Ms. Valeria Maria Barros de Andrade for sparing her time to specify the Statecharts model for both the examples and for running the program to obtain the results.

References

1. ANDRADE, V.M.B., CARVALHO, S.V., VIJAYKUMAR, N.L Desenvolvimento de um Software para análise de desempenho de sistemas através de modelos markovianos. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 29, 1997, Salvador, Brasil. Anais dos Resumos... Salvador, Brasil: SOBRAPO, 1997.
2. ATTIEH, A., BRADY, M.C., KNOTTENBELT, W.J., KRITZINGER. (1995, May). Functional and Temporal Analysis of Concurrent Systems. http://www.cs.uct.ac.za/~william/DNAnet.html.
3. HAREL, D. Statecharts: a visual formalism for complex systems. Science of Computer Programming, v. 8, p. 231-274, 1987.
4. Harel, D., PNUELI, A., SCHMIDT, J., SHERMAN, R. On the formal semantics of Statecharts. In: IEEE SYMPOSIUM ON LOGIC IN COMPUTER SCIENCE, 1987, Ithaca. Proceedings... Ithaca, USA: [s.n], 1987.
5. HAREL, D., LACHOVER, H., NAAMAD, A., PNEULI, A., POLITI, M., SHERMAN, R., SHTULL-TRAUTIN, A., TRAKHTENBROT, M. STATEMATE: A Working Environment for the Development of Complex Reactive Systems. IEEE Transactions on Software Engineering, v. 16, n. 4, p. 403-414, Apr. 1990.
6. HAREL, D., NAAMAD, A. The STATEMATE Semantics of Statecharts. ACM Transactions on Software Engineering, v. 5, n. 4, p. 293-333, Oct. 1996.
7. KLEINROCK, L. Queueing Systems. v. 2, New York, USA: John Wiley & Sons, 1976.
8. MURATA, T. Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE, v. 77, n. 4, p. 541-580, 1989.
9. PHILIPPE, B., SAAD, Y., STEWART, W.J. Numerical Methods in Markov Chain modeling. Operations Research. v. 40, n. 6, p. 1156-1179, 1992.
10. SILVA, E.A.S., MUNTZ, R.R. Métodos Computacionais de Solução de Cadeias de Markov: Aplicações a Sistemas de Computação e Comunicação. Gramado, Brasil:. Instituto de Informática da UFRGS, 1992.
11. SILVA, A R. F. Modelos de redes de filas para Sistemas Computacionais Distribuídos-Simulação X Métodos Analíticos. São Carlos, ICMC-USP, 2000, (M.S. dissertation)
12. VIJAYKUMAR, N.L., CARVALHO, S.V., ABDURAHIMAN, V. On proposing Statecharts to specify Performance Models. International Transactions in Operational Research, 1999. (Submitted).
13. VIJAYKUMAR, N.L. Statecharts: Their use in specifying and dealing with Performance Models. São José dos Campos, ITA, 1999, (D.S. thesis).

NOTAS DO ICMC

SÉRIE COMPUTAÇÃO

050/2000 FARIA,G.; ROMERO, R.A.F. – Incorporating fuzzy logic to reinforcement learning.

049/2000 OLIVEIRA, P.R.; ROMERO, R.F.; NONATO, L.G.; MAZUCHELI, J. – Techniques for image compression: a comparative analysis.

048/2000 LINS, L.; LINS, S.; MORABITO, R. – Na n-tet graph approach for non-guillotine packings of n-dimensional boxes into na n-container.

047/2000 OLIVEIRA JR., O. N.; MARCHI, A.R.; MARTINS, M.S.; MARTINS, R.T.; NUNES, M.G.V. – A critical analysis of the performance of english-portuguese-english MT systems.

046/2000 SOUZA, P.S.L.; SANTANA, M.J.; SANTANA, R.H.C. – Escalonamento de processos: uma contribuição para a convergência da área.

045/2000 MARTINS, R.T.; RINO, L.H.M.; NUNES, M.G.V.; MONTILHA, G.; OLIVEIRA JR., O. N. – An interlingua diming at communication on the Web: how language-independent can it be?

044/99 FELTRIM, V.D.; FORTES,R.P.M. – Evaluation of a reverse engineering method through the application in a hypermedia system.

043/99 PANSANATO, L.T.E.; NUNES, M.G.V. – EHDM: Método para projeto de hiperdocumentos para ensino.

042/99 MORABITO, R.; ARENALES, M. – Optimizing the cutting of stock plates in a furniture company.

041/98 SANTOS-MEZA, E.; SANTOS, M.O.; ARENALES, M.N. – Lot sizing and scheduling in na automated foundry.