# On the use of UML State Machines for Software Performance Evaluation*

José Merseguer

Dpto. de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, Spain
jmerse@unizar.es

**Abstract.** Being a goal of the workshop to suggest improvements to the UML profile for SPT, this paper tries to point out that UML state machines can be a useful tool to model performance requirements. The proposal presented is based on some experiences on the use of the UML SMs with performance evaluation purposes.

## 1 Introduction

A software retrieval system and a mail protocol were respectively modeled and analyzed in [5, 7]. In those works, we used UML state machines (SMs) to describe the behavior of objects making up the system. Then, we realized that UML SMs are capable to represent some performance requirements. Moreover, analyzable performance models were derived in terms of GSPNs [1].

In this paper are identified some performance requirements that UML SMs are able to represent. Also an interpretation on the use of the UML SMs for performance evaluation is given. Obviously, the experiences we have recorded do not take into account all the potential that UML SMs have for performance evaluation, so we just highlight the points that we have dealt with.

Section 2 recalls some basic features of the UML SMs. Section 3 suggests some points that could be taken into account to use UML SM for performance evaluation.

## 2 UML State Machines

UML State Machines offer a set concepts fairly adequate for the modeling of discrete behavior. Their capability to model *object reactive behavior* turns them into an adequate tool to model different aspects of real time systems. In our approach, SMs are used to model the life-cycle of the objects, the so-called *executable protocol SMs* in UML 2.0. UML SMs are proposed as a variant of Harel statecharts [2].

"A state in a SM models a situation during which some invariant condition hold. The invariant may represent a static situation, such an object waiting for an external event to occur, or a dynamic one such performing some activity. A simple transition is a relationship between two states indicating that an object in the first state will enter the second state; in protocol SMs, they have not associated an *effect*. A transition can have a *trigger*, meaning that an event may cause the execution of an associated behavior" [8].

Our experience on the use of the UML SMs for performance evaluation concerns with the modeling and analysis of software systems [5, 7]. We made use of the 1.4 version of the

* This work has been developed within the projects: Performance from Unified Model Analysis (PUMA) funded by the NSERC of Canada and TIC2003-05226 of the Spanish Ministry of Science and Technology.

UML. Another works about the use of UML SMs pointed them as an adequate tool for the study of the QoS characteristics of a system [3]. In [4], a tool was developed to automatically verify systems, described as a set of UML SMs, using the SPIN model checker.

UML 2.0 has introduced several changes in the definition of the UML SMs package with respect to the version 1.4. Despite this proposal has not been updated on the new version, we devise that two of the changes could be relevant: The change of the Action class for the Activity class; and the semi-formal definition of the *UML protocol SMs*. Neither the rest of the changes in the SMs metamodel seem to affect our proposal nor they seem to be of great importance for performance evaluation.

## 3 UML State Machines and the SPT

Since the behavior of a system can be modeled as a set of UML SMs, actually the ones that describe the life of each of the involved objects, it should be possible to obtain a performance model from these diagrams. In [6] was presented an approach, where performance models in terms of stochastic Petri nets [1] can be obtained starting from UML SMs.

In this section is proposed an approach to model performance requirements using UML SMs. The approach is just based on our modeling and analysis experiences [5, 7] using UML SMs. It means that a lot of possibilities still remain open.

In subsection UML viewpoint are suggested the mappings of the proposal, from the performance concepts into UML equivalents. After, the required extensions are given.

### 3.1 UML viewpoint

An approach based on SMs could be useful for engineers that used to model systems by describing the behavior of cooperating objects.

It will be assumed that each class in the system with relevant dynamic behavior is modeled using a SM.

***PerformanceContext***

Since a performance context contains one or more scenarios that represent responses with response times and throughputs, two kind of performance contexts could be devised:

A) The Performance Context of the whole system.

Assuming that all of the SMs together represent the behavior of the whole system, then all of them together could define a performance context, where to explore all the dynamic situations in the system and where to compute performance measures for the system as a whole.

In the context of the SPT a problem arises since some model element should be stereotyped ≪PAcontext≫ , i.e., what is the "Base Class" for the stereotype ≪PAcontext≫ ? If the behavior of the system is represented by only one SM (a special case), then the stereotype could be applied to that SM, being StateMachine that "Base Class".

B) The Performance Context represented by a SM.

Each SM, making up the system, could be a performance context by itself. Since a SM represents a pattern of behavior for a class of objects, then it can be interpreted as the performance context (i.e. scenario) where such objects execute. In this case, the performance interpretations could be quite similar to the "Activity-Based Approach" given in the SPT, but excluding those elements typical of the Activity Diagrams such as swimlanes.

As an example among the infinite interpretations that the *response time* attribute could take, one typical could be to measure the lifetime of the object.

***Step***

The model elements of a SM that take time to execute, and/or represent an increment in the execution, and/or use resources are: doActivities, transitions and events.

*doActivities* represent tasks (whose computation time should be annotated) performed by an object in a given state. Be aware that it is not proposed as a Step other kinds of Actions, i.e., entry, exit or the effect in a transition.

In *transitions*, guards (routing rates) show conditions that must hold in order to fire the corresponding event. Then the probability of guard success could be annotated in the transition.

*Events*, labeling transitions, correspond to events in a sequence diagram showing the server and client sides of the object. In the case of objects executing in the same machine (it has to be modeled in the deployment diagram) can be considered that the time spent to send the message is not significant in the scope of the modeled system. But for those messages traveling through the net, the size of the message can be annotated in the transition. If the speed of the net is annotated (in the deployment), then the delay in the system, generated by the message, could be calculated.

***Workload***

A workload in SPT is associated to a scenario, then in this approach mainly to a SM. A SM represents a pattern of behavior for a class of objects, but performance analysis is inherently instance-based. So, it can be interpreted that each object populating the class has its own copy of the SM.

The number of objects populating the class can be specified in the population attribute of the class ClosedWorkload, then indicating how many replicas exist for the SM.

***Resources***

As far as we have experienced with them, we consider the deployment diagram the place to specify the location of the objects and the speed of the net.

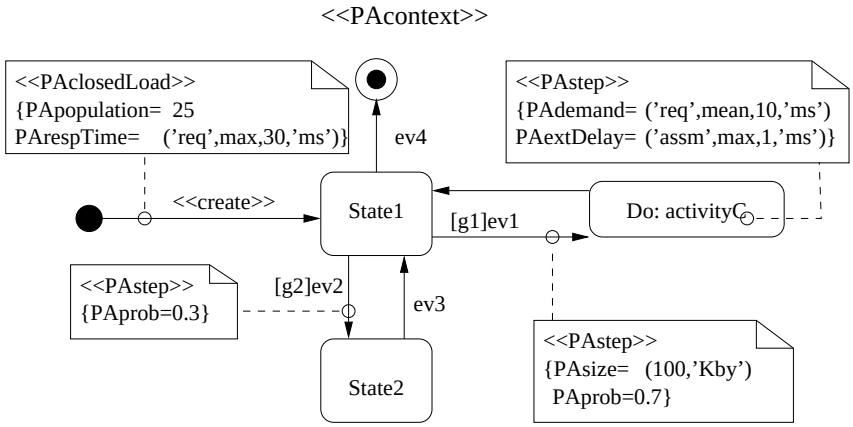


**Fig. 1.** State Machine with performance annotations.

### 3.2 Extensions required

The extensions required to deal with UML SMs under this perspective are given in table 1. Figure 1 depicts a SM with the proposed annotations.

| **Stereotype** | **Base Class** | **Tags** |
|---|---|---|
| ≪PAcontext≫ | (1) | |
| ≪PAstep≫ | Transition (2) | PAsize<br>PAprob |
| | Activity (role doActivity) | (3) |
| ≪PAclosedLoad≫ | Transition | PArespTime<br>PApopulation |

(1) Base Class could be StateMachine when the PerformanceContext is represented by only one SM.
(2) Transition is proposed as a special kind of Step, that has only 2 attributes: size and probability.
(3) All the tags of a ≪PAstep≫.

**Table 1.** Extensions.

## 4 Conclusion

UML SMs can be a useful tool to model real-time systems since they are adequate to describe reactive behavior. If they are added to the SPT, some challenges in this kind of systems could be addressed.

## References

1. M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with generalized stochastic Petri nets*, John Wiley, 1995.
2. D. Harel, *Statecharts: a visual formalism for complex systems*, Science of Computer Programming **8** (1987), no. 3, 231–274.
3. D.N. Jansen, H. Hermanns, and J.P. Katoen, *A QoS-oriented extension of UML statecharts*, UML 2003, LNCS, vol. 2863, Springer, pp. 76–91.
4. J. Lilius and I. Porres-Paltor, *vUML: A tool for verifying UML models*, ASE'99, IEEE Computer Society, pp. 255–258.
5. J.P. López-Grao, J. Merseguer, and J. Campos, *Performance engineering based on UML and SPNs: A software performance tool*, ISCIS XVII, CRC Press, pp. 405–409.
6. J. Merseguer, *Software performance engineering based on UML and Petri nets*, Ph.D. thesis, University of Zaragoza, Spain, March 2003.
7. J. Merseguer, J. Campos, and E. Mena, *Analysing internet software retrieval systems: Modeling and performance comparison*, Wireless Networks **9** (2003), no. 3, 223–238.
8. Object Management Group, `http:/www.omg.org`, *OMG Unified Modeling Language specification*, August 2003, version 2.0.