SIMULATION MODELS VALIDATION USING THE OSSIM TOOL

Elena ULEIA[1]

Obiectul lucrării de față este prezentarea a doua metode de validare a modelelor de simulare, dezvoltate folosind pachetul de programe OSSim (Open Source Simulator). Pentru a adresa această problemă, s-a utilizat un sistem de cozi. Ambele metode de validare folosesc entități 'observer', care evaluează anumite propietăți ale sistemului de simulare, verifică corectitudinea datelor analizate si le raportează. Metodele prezentate validează în mod static modelul de simulare. Această prezentare arată suportul simulatorului OSSim pentru validarea modelelordezvoltate in acest mediu.

The objective of this paper is to present two validation methods for simulation models developed using the OSSim (Open Source Simulator) tool. In order to address this matter, a simulation model of a queueing system has been used. Both validation methods make use of observer entities, which evaluate certain simulation system properties, check the correctness of the analyzed performance data, and report the results. The present methods are statically validating the simulation model. This proves the OSSim tool capability for validation purposes.

Keywords: simulation, validation, queuing systems, hierarchical models, performance analysis, distributed computing

1. Introduction

The present paper introduces two validation methods based on observers. The validation is performed for a queueing system simulation model in order to check the model validity. Both methods used in this paper are statically validating the model. These can be extended to a dynamic validation by using a runtime check on the system properties.

As discussed by [1], a set of 10 practical techniques for developing valid and credible models are described, including: quantitative techniques, performing sensitivity analyses to determine important model factors, or validating the output from the overall simulation model.

It is usually too difficult, too expensive, or too time consuming to use all possible validation techniques for every model that is developed. The personal contribution relies on the OSSim tool design that best suits fast model building an

[1] PhD student, Computer Science Faculty, University POLITEHNICA of Bucharest, Romania, elena.uleia@gmail.com

easy extension for validation support. OSSim tool supersedes other simulator tools of the same functionality levels by its fast runtime execution [2].

2. OSSim Tool

OSSim [3] is a discrete event simulator package that allows for development of network simulation and analysis which makes use of technologies from the area of distributed computing, client-server architectures and object oriented programming. One of the key issues of the design is high performance of the overall system, making possible the modelling and the simulation of complex networks.

OSSim is also suitable for simulation of any system with a hierarchical structure which admits a discrete time modelling.

3. Simulation model description

The system chosen for modelling, validation, and simulation, using the OSSim toolkit, is a queueing network. The system consists of a network with three uni-servers, each of these having different number of inputs and outputs itself, but only one queue per server (uni-server), two incoming external packets traffic lines, and two lines for outgoing traffic.

The topology of the queueing network is drawn in fig. 1 below.

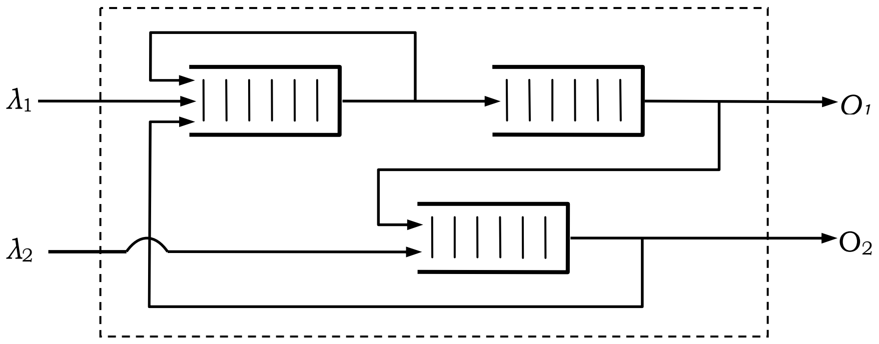

Fig. 1. Queueing Network System

The queues of each uni-servers are sufficiently large (10^6 bits), i.e. no loss of packets is encountered in the system for the specified bit rate. The bit rate for all queues is assumed identical, of 33,600 bits/sec. The uni-server queue discipline is FIFO, first arrived packet is the first one served.

The input traffic lines support packets of different length, following a uniform distribution, between 100 and 900 bits. The packets arrival rate (λ_1, λ_2) follows an exponential distribution with the mean value of 1 sec. The selection probability of the input traffic lines follows a Bernoulli distribution with probability 0.4.

4. Simulation model validation

The simulation model is viewed as a ‘black box’ which takes all input variable specifications and transform them into a set of output or response variables [4]. Figure 2 shows the input and output variables along with the ‘black box’ model, for our queueing simulation system.

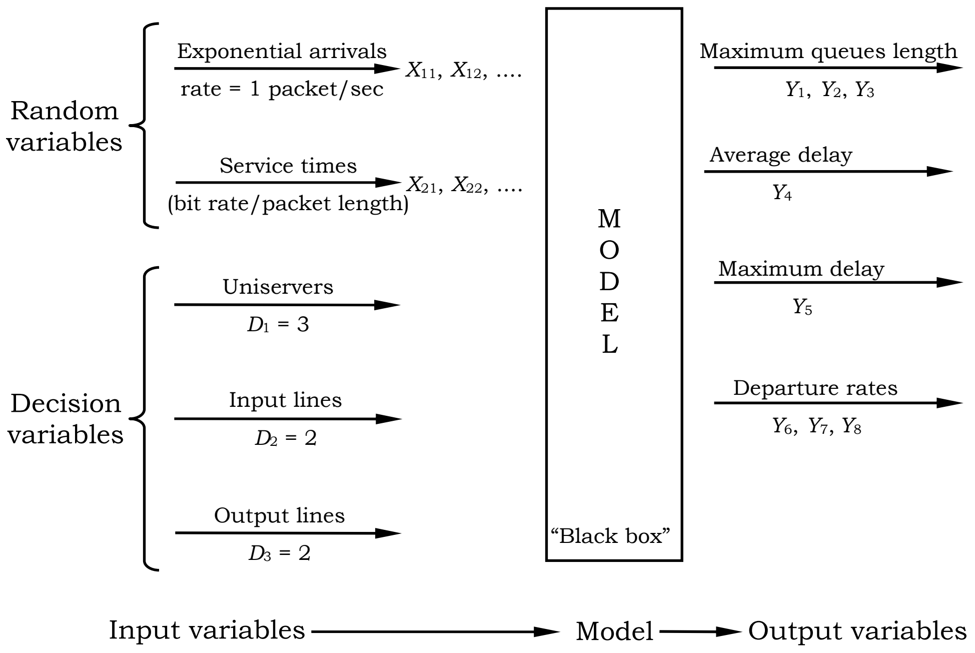

Fig.2. Model input-output transformation

The output variables consist of all statistics of interest generated by the simulation with respect to the models’ behaviour. For example, one may be interested in the average delay in seconds of a customer (i.e. packet) from arrival to beginning of service, or the maximum length of queue size at pick times.

The uncontrollable input variables are denoted by X, the decision variables by D, and the output variables by Y. from the ‘black box’ point of view, the model takes the inputs X and D and produces the outputs Y:

$$\mathrm{f}\,(X, D) = Y$$

here, f denotes the transformation that is due to the structure of the model.

For our example model, the exponentially distributed inter-arrival time generated in the model between packet n-1 and n is denoted by X_{1n}. The normally distributed service time generated in the model for packet n is denoted by $Y_{1n.}$

For validation of the input-output transformations of the queues model to be possible, real system data must be available, comparable to at least some of the model output *Y*. As our example is an ideal model for a network of queues, it is unlikely that such results could be produced.

5.Validation techniques for conventional simulation models

A taxonomy of more than 77 V&V (Validation & Verification) techniques for conventional simulation models is presented in Fig. 3. Most of these techniques come from the software engineering discipline and the remaining are specific to the modelling and simulation field. Details of these techniques can be found in [5]. The V&V techniques are classified into four primary categories: informal, static, dynamic, and formal.

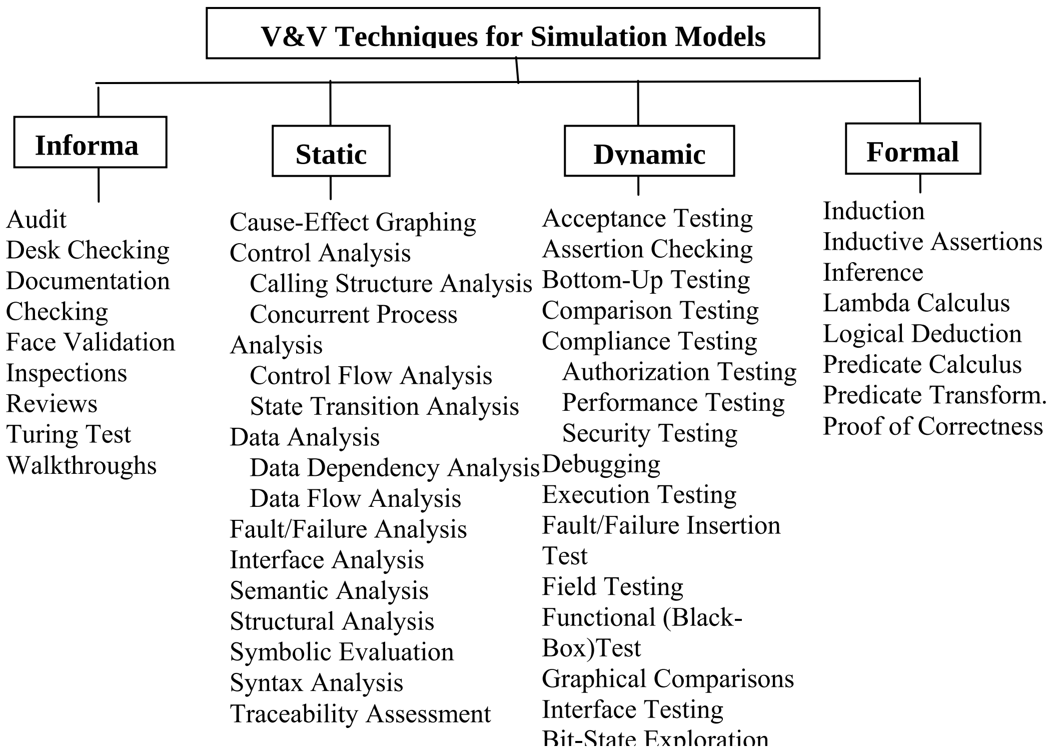

Fig. 3. Classification of Verification and Validation Techniques for Conventional Simulation Models

Informal techniques are among the most commonly used. They are called informal because the tools and approaches used rely heavily on human reasoning and subjectivity without stringent mathematical formalism. These techniques are applied using well structured approaches under formal guidelines and they can be very effective if employed properly.

Static techniques are concerned with accuracy assessment on the basis of characteristics of the static model design and source code. Static techniques do not require machine execution of the model, but mental execution can be used. The techniques are very popular and widely used, with many automated tools available to assist in the V&V process. These techniques can obtain a variety of information about the structure of the model, modelling techniques and practices employed.

Dynamic techniques require model execution and are intended for evaluating the model based on its execution behaviour. Most dynamic V&V techniques require model instrumentation. Dynamic V&V techniques are usually applied using the following three steps. In Step 1, the executable model is instrumented. In Step 2, the instrumented model is executed and in Step 3, the model output is analyzed and dynamic model behaviour is evaluated.

Formal techniques are based on mathematical proof of correctness. If attainable, proof of correctness is the most effective means of model V&V. Unfortunately, “if attainable” is the overriding point with regard to formal V&V techniques. Current state-of-the-art proof of correctness techniques are simply not capable of being applied to even a reasonably complex simulation model. However, formal techniques serve as the foundation for other V&V techniques.

6. Validation by observers

Discrete-event systems can be formalized by using finite automata over a set of *observable* events plus a set of *unobservable* events [6].

Any run is a sequence of states; a trace is a sequence of observations on a subset of the variables. These variables can represent input and output ports, while the internal variables would be hidden from an observer. An observer will see the input values at the same time as the output values, and will not be able to infer causality relations without additional information [7].

The validation process will be actually performed by introducing *observers* in the system, in order to collect model properties at the *observer* check point, and compare these with analytically derived model properties.

In the remaining part of this paragraph are given two observers validation patterns. First one checks the queue departure rate, and the second one the cumulative queue arrival rates.

6.1. Validation of departure rate

The simplest validation property for a network of queues is the first property in [4], that states:*' Provided no customers are created or destroyed in the queue, then the departure rate out of a queue is the same as the arrival rate into* *the queue, over the long run.'* For this purpose a new *observer* block has been created (*my_obs*), with an attached *observer* process. The observed property, the departure rate, is checked statically for queue q32. All of the inter-departure times are collected and saved in a log file for post-processing after simulation is complete. The validation queues model is shown in fig. 4 below.

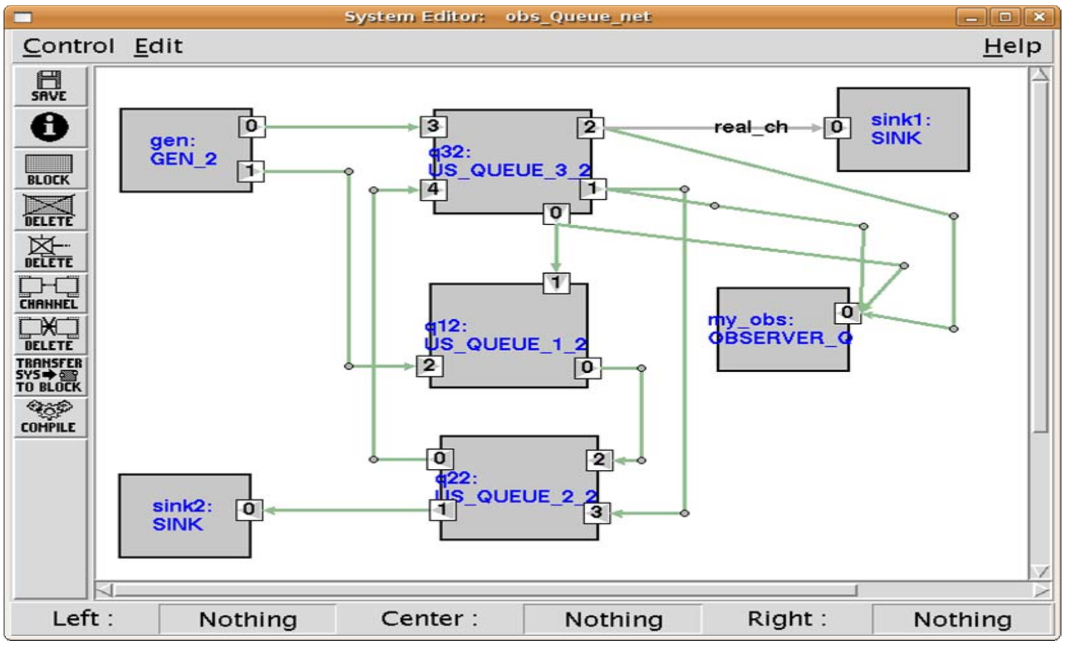

Fig. 4. Model Validation Block Diagram

All simulation runs here were performed with the same simulation parameters as the runs for the simulation model, i.e. without any observers attached. The results for the inter-departure times are reproduced in the table 1 below, in *us_queue_q32_out*.

Table 1

Confidence Intervals for 90%, 95% and 99% confidence levels

CI (Confidence Intervals)	90%	95%	99%
us_queue_q12	1.187003-1.203262	1.185302-1.204963	1.181725-1.208540
us_queue_q22	0.928135-0.941438	0.926743-0.942830	0.923816-0.945757
us_queue_q32	1.057704-1.071656	1.056244-1.073116	1.053174-1.076185
us_queue_q32_out	1.057697-1.071663	1.056236-1.073124	1.053163-1.076196

The confidence intervals for both queue q32 input and output parameters are following closely together, with a minor variance of 10^{-4} %.

The queue q32 inter-arrival times mean converges to a stable state, after initial transitory state, at about 1500th sample. The q32 inter-departure times mean (red line in the figure 3 graph) converges at the same sample point.

The queue observed output parameter following closely the input parameter is proved both graphically (fig. 5), and analytically by means of the confidence intervals results (table 2).

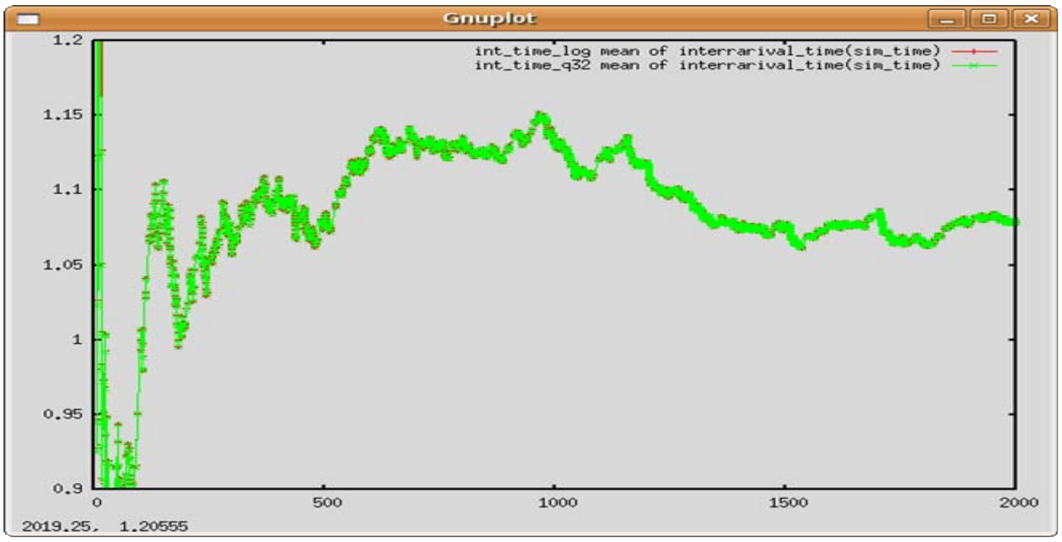

Fig. 5 Model Validation – Output/Inputs Rates for q32

6.2. Validation of network queues convergence

This involves validation of the stability (convergence) system property. The property that is validated here involves the property that states: '*If customers arrive to queue i at rate λi, and a fraction 0 < pij < 1 of them are routed to queue j upon departure, then the arrival rate from queue i to queue j is λi * pij over the long run*' [4]. Additionally, there has to be check that all packets that enter the network queues system are leaving the queueing system, i.e. no packets loss.

In order to prove the stability of the whole system, the following convergence equation should hold for each queue in the system: inter-arrival mean time value at queue *i*, multiplied by the ratio of the number of packets received at current queue *i* ports and total number of the generated packets should equal the mean value of the generated packets in the system.

$$\lambda = \lambda i \frac{L_i}{L} \tag{1}$$

where, λ – is the mean arrival time of the generated packets;

λ_i – inter-arrival mean time at queue *i* ;

L – total number of generated packets injected in the system during simulated time T;

L_i – number of packets arriving at queue *i* input ports, during simulated time T.

For this purpose, two *observer* blocks have been added to the system (*obs1, obs2*), with an attached *observer* process each. The observed property, is checked statically for each queue. All of the inter-arrival times are collected and saved in a log file for post-processing after simulation is complete.

The validation queues model is shown in fig. 6 below.

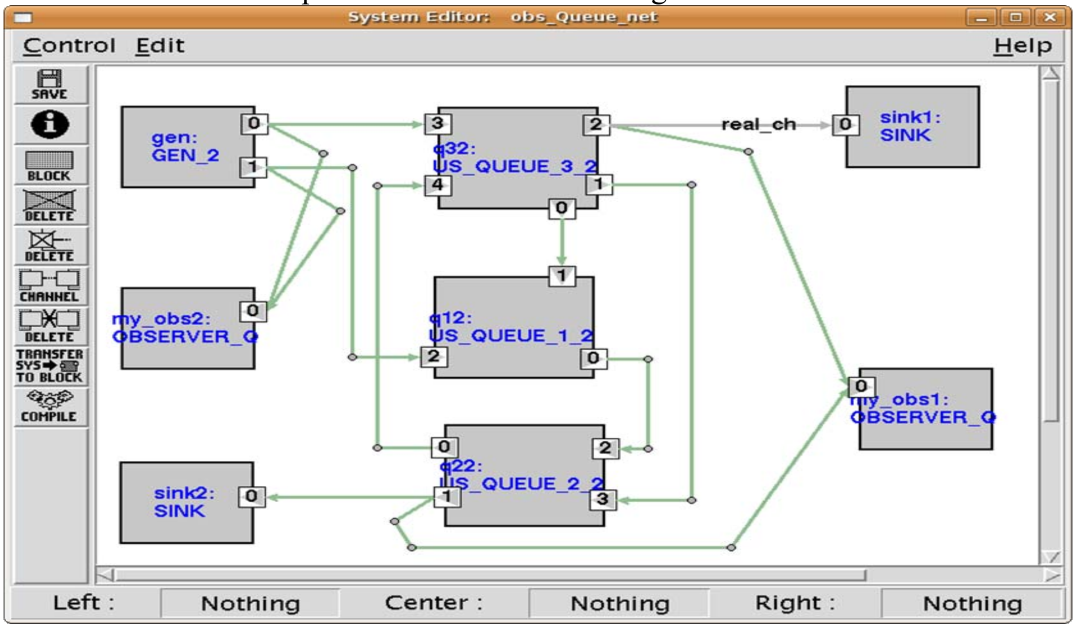

Fig. 6. Model Validation Block Diagram

All simulation runs have been performed with the same simulation parameters as the runs for the simulation model, i.e. without any observers attached. The results for the inter-arrival and inter-departure times are reproduced in the table 2 below, in us_queue_obs1, and us_queue_obs2 respectively.

Table 2

Confidence Intervals for 90%, 95% and 99% confidence levels

CI (Confidence Intervals)	90%	95%	99%
us_queue_q12	1.197926-1.202556	1.197441-1.203041	1.196422-1.204059
us_queue_q22	0.935417-0.939205	0.935021-0.939602	0.934187-0.940435
us_queue_q32	1.070029-1.075934	1.069411-1.076552	1.068112-1.077851
us_queue_obs1	0.998092-1.001339	0.997752-1.001679	0.997037-1.002394
us_queue_obs2	0.998070-1.001361	0.997726-1.001705	0.997002-1.002429

The queue *obs1* (input to the system) and *obs2* (output from the system) inter-arrival times mean (green and red lines in the fig. 7 graph) converge to a stable state, after initial transitory state.

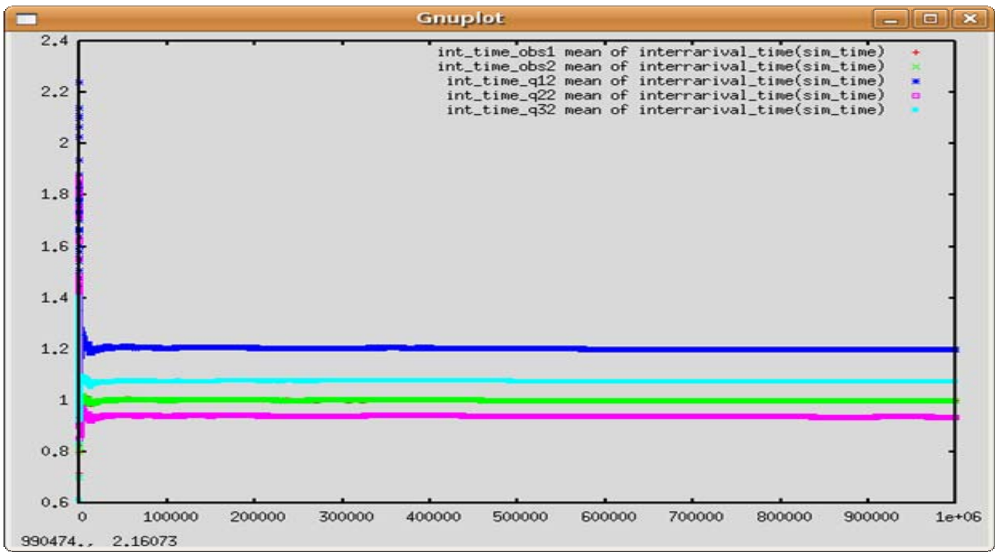

Fig. 7. Model Validation – Output/Inputs Rates for *q12, q22, q32, obs1*, and *obs2*

The queues inter-arrival mean times parameters are proved both graphically (figure 6), and analytically by means of the equation (1) calculations, in table 3 below.

Table 3

Mean inter-arrival times - computed and experimental values

Convergence results	No of samples of queue *i* [N_i]	Mean of interarrival time of queue *i* [λ_i]- computed	Mean of interarrival time of queue *i* [λ_i]- experimental
us_queue_q12	833155	1.200590	1.200241
us_queue_q22	1066881	0.937572	0.937311
us_queue_q32	931978	1.0732849	1.0729815
us_queue_obs1	1000278	1	0.9997155
us_queue_obs2	1000278	1	0.9997155

From the above table, the q12 received packets are 833155+1, the total generated packets number is 1000278+1, and the generated packets distribution mean value is 1. By applying equation (1), the queues inter-arrival mean time is:

$$\lambda = \lambda i \frac{L_i}{L} \quad => \quad \lambda i = \lambda \frac{L}{L_i} \qquad (2)$$

$$\lambda 1 = \lambda \frac{L}{L_1} = 1 * 1000279 / 833156 = 1.200590$$

$$\lambda 2 = \lambda \frac{L}{L_2} = 1 * 1000279 / 1066882 = 0.937572$$

$$\lambda 3 = \lambda \frac{L}{L_3} = 1 * 1000279 / 931979 = 1.073284$$

The experimental results are very close to the computed results, which prove the convergence of the system. For the current simulation time, that is 10^6, the convergence deviation is about 2.8 10^{-4} for all computed inter-arrival mean times. A null convergence deviation is to be obtained for very long simulation time. For example, a simulation run of 10^8-10^9 is considered sufficient for the current queueing system.

7. Conclusions

Model validation is a mandatory requirement in all simulation models design. The validation methods presented in this paper are making use of the observers concept that is applied in validating a queueing simulation model developed using the OSSim network simulation environment.

The OSSim tool is capable of simulating, on usual hardware, 1.2E10 events within one hour, based on several implementation optimisations [8].

Other experiments are planned to be performed, with the same simulation model, but having the system property checked dynamically, instead. The correctness requirement check will be done at runtime, and any failure reported by a corresponding error message. This method is checking the validity of the model faster than the static one, improving the tool validation performances.

REFERENCES

[1]. *Robert G. Sargent,* 'Verification and Validation of Simulation Models', Proceedings of the 2005 Winter Simulation Conference (December 2005, Orlando, FL, USA), pp.130-143, 2005.

[2]. *Elena Uleia,* 'Performance evaluation for Discrete event Simulators: OSsim versus OMNeT++', U.P.B., Scientific Bulletin, No.3, 2008

[3]. *Elena Uleia, O. Fratu, & Simona Halunga*, Discrete Event Simulator for Communication Networks', 15th Telecommunications forum TELFOR 2007, Serbia, Belgrade, Nov. 20-22, 2007.

[4]. *J. Banks, J.S.Carson II, B.L. Nelson*, Discrete-Event System Simulation, Prentice Hall, 1999

[5]. *O. Balci*, 'Verification, Validation and Accreditation of Simulation Models', Proceedings of the 1997 Winter Simulation Conference, pp 135-141.

[6]. *A.M. Law*, 'How to Build Valid and Credible Simulation Models', Proceedings of the 2005 Winter Simulation Conference, M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds., December 2005, Orlando, FL, USA), pp 24-31, 2005.

[7]. *J.C.P. Kleijnen*, 'Validation of Models: Statistical Techniques and Data Availability', Proceedings of the 1999 Winter Simulation Conference, pp 647-653.

[8]. *Elena Uleia, Simona Halunga*, Implementation techniques for communications protocols, International Symposium on Signals, Circuits and Systems, 2005, ISSCS 2005, **Vol. 2**, 14-15 July 2005 Page(s):529 - 532