AMO - Advanced Modeling and Optimization, Volume 4, Number 3, 2002

UPON VARIABLE NUMBER OF WORKING STATIONS PREEMPTIVE QUEUING SYSTEM: A FORMAL AND OBJECT ORIENTED APPROACH

ION FLOREA
" Transilvania " University of Brasov
e-mail:i.florea@info.unitbv.ro

Abstract

Complex queuing systems imply a discipline of services based on priority treatment of certain clients. The preemptive queuing system type is an appropriate model for numerous real systems. In [6] I have introduced the variable number of parallel working stations queuing system and in [5] I have presented a simulation algorithm for the preemptive queuing system of this type. Based on the DEVS formalism, in [4], respectively in [8], I have been proposed formal and object oriented models, for preemptive queuing system with one station, respectively with a constant number of parallel working stations. Using the same general framework, in this paper I propose a simulation model for a variable number parallel working stations preemptive queuing system. Formal and modular description of systems behavior eliminates the possibility of introducing errors. To each DEVS model, an abstract simulator we associate and using the advanced concept of Object Oriented Language C++, the abstract simulator was implemented.

Key Words: Simulation, Queuing Models, Variable Number of Working Stations, Object Oriented Design, DEVS Model

AMS Classification: 60K25, 65C20, 68U20

1 Introduction

If m is the priority class number, then the i-class clients have serving priority versus those of j-class if $i < j$; $i, j = 1, ..., m$. The constant number parallel servers head-of-line systems, one bases on the following rules: Inside the same class the customers are served in the *first-come-first-served* order; when a customer is arriving and there is a server in laziness, the customer is immediately served, otherwise, it is last queued inside his class; when a customer is arriving and there is a server in laziness, the customer is immediately served, otherwise it is last queued inside his class; on a high priority arrival versus the priorities of the queued customers, the arrived customer wait until one serving finishing, in order to be himself served; when one server finishes to serve a customer, if the queue is not empty, the first queued of the high class is selected to be served; this rule stands out the priority customers serving.

In the preemptive parallel multi-server queuing system case, in the situation of arriving a higher priority class customer versus the priorities of the serving clients, one of the customers of the low class is preempted, and it joins on the first place inside his class. His new serving time can be the older service time (the service resumption at begin) or the remaining time, until the service completion (the service resumption from the remaining point). We consider the last case, because it is more general. So, any client from the queue may be preempted or not. We will stand out this fact, considering a variable "tc" (client type) associated to any client from the queue; so, *tc=1*, if the respective client is preempted, otherwise *tc=0*.

There are situations when the number of the servers can vary, depending on the length of the queue. So, if the number of the queued customers is sufficiently of great, in order to decrease the customers waiting time, the adding in use of a new server is necessary; conversely, if the number of the queued customers is sufficiently of small, in order to decrease the laziness time of the servers, the removing from use of the servers is necessary. Naturally, it is necessary to introduce the adding, respectively the removing criterion. So, the adding of a new server criterion is $ql/nbp \geq lmax$, respectively, the removing criterion will be $ql/nbp > lmin$, where ql is the number of the queued customers, nbp is the active servers number and $lmax$, $lmin$ are two model parameters. Also, we assume that at any moment, there is at least one server in use, just if the queue is empty; in this case, the server is in laziness and it is waiting in order to serve a possible arrival.

The difference between the constant parallel servers model and that with variable number of servers, consists in the server laziness time establishing. Because, in the first case, one server that not works is in laziness, in the second only when the queue is empty, the server that is waiting a possible arrival is in laziness. The simulation methods of these systems are important, because there are not analytic formulae in order to study these models.

2 Formal, modular and hierarchical construction of simulation models

Modeling and simulation of a system implies three basic objects: the real system, the model and the simulator. Thus a model is defined by:

$$M = (X, Y, S, \delta_{int}, \delta_{ext}, ta, \lambda) \tag{1}$$

where: X is the set of external event types; S is the set of states, s^0 is the initial state; Y is the set of external event types generated as outputs; $\delta_{int} : S \to S$ is the internal transition function, which dictates the transition of the model in the absence of external events; $ta : S \to R_+ \cup \{0, \infty\}$ is the time advance function, i.e. if the model is in state s, it will reach state $\delta_{int}(s)$ after a time $ta(s)$;

$\delta_{ext} : S \times Q \to S$, where $Q = \{(x, e) / x \in X, e \in R_+ \cup \{0, \infty\}, 0 \leq e \leq ta(s)\}$, is the external transition function, $\delta_{ext}(s, x, e)$ is the state of the model that it will reach after time e, if it has received an input x; $\lambda : S \to Y$ is the output function, $\lambda(s)$ is the output generated on internal transition from state s.

Object-oriented modeling and simulation supposes: modular construction of the models associated to the systems; specification of the models linking rules; the simulator associated to each model is an object, related to a programming language; the communication between objects is achieved by message transmission technique. The connection of the DEVS models is achieved by employing the model concept with several components or the fixed structure network, defined by the 5 -uple:

$$DN = (D, \{I_i\}, \{M_i\}, \{Z_{i,j}\}, select) \tag{2}$$

where D is the set of the component models identifiers; for each $i \in D : \{M_i\}$ is a basic model component; $\{I_i\}$ is the set of the influenced models of the i component; for every $j \in I_i$ we define $\{Z_{ij}\}$ the message transmission function from model i to model j; *select* is the separation function by which one component is selected of several who change their state at a given moment. Further, we define the **atomic model** as being a basic model which is not obtained by connecting other models. The basic multi-component modeling in the DEVS formalism context is the fact that any such model is also a DEVS model. This result is proved in [17], and based on the hierarchical construction of the complex model is possible.

The experimental frame concept has been introduced in order to separate the inputs, outputs and the model dynamic [17, 12, 19]. The inputs, respectively the outputs, correspond to generator model, respectively to transducer model. So, considering one model it is possible to change the inputs mechanism or to change the output variables, in order to establish other model efficiency factors.

The support for modelling variable structure systems is provided by the DSDEVS formalism [1]. In the context of this theory the models are of two types: **basic**, respectively **network** models. A DSDEVS basic model is a DEVS model. The network model is a combination of DSDEVS basic models. $DSDEVN_\Delta = (\chi, M_\chi)$ defines the DSDEVS dynamic structure network model; Δ is the name of the DSDEVS network, χ is the name of the DSDEVS network directory and $M_\chi = (X_\chi, Y_\chi, S_\chi, \delta_{int_\chi}, \delta_{ext_\chi}, ta_\chi, \lambda_\chi)$ is the basic model of χ. The information about the network dynamic structure is located in the state of the executive. A state $s_\chi \in S_\chi$ of the directory is defined by: $s_\chi = \left(X_\Delta^\chi, Y_\Delta^\chi, S_\Delta^\chi, D^\chi, \{M_i^\chi\}, \{I_i^\chi\}, \{Z_{i,j}^\chi\}, select^\chi, \theta^\chi\right)$, where X_Δ^χ is the set of input events into the DSDEVS network; Y_Δ^χ is the set of output events from the DSDEVS network; D^χ is the set of components; I_i^χ is the set of components influenced by i, $i \in D^\chi$; $Z_{i,j}^\chi$ is the translation function of messages from component i to component j, $j \in I_i^\chi$; $select^\chi$ is the selection function; θ^χ means other variable, not previously defined. The 7-uple $\left(X_\Delta^\chi, Y_\Delta^\chi, S_\Delta^\chi, \{M_i^\chi\}, \{I_i^\chi\}, \{Z_{i,j}^\chi\}, select^\chi\right)$ is called the network structure.

Any change in any one of these variables is called change of structure, i.e.

adding or removing components, modifications links between components. The network structure can change when the directory performs an internal or external transition. Thus as all structure information is kept in the network directory state, any such change will be accompanied by directory transitions. In [1] has been proved that a DSDEVS dynamic structure network model is a DEVS model. This guarantees that by hierarchical construction of the variable structure network models, also a basic model is obtained.

3 The head of line variable number of working stations queuing system DEVS study

3.1 The model static structure

The first step in the hierarchical construction of the simulation models is to identify the atomic model components and the connecting relations between different models. This information may find in the **digraph model**[17, 18] that is presented in figure 1.

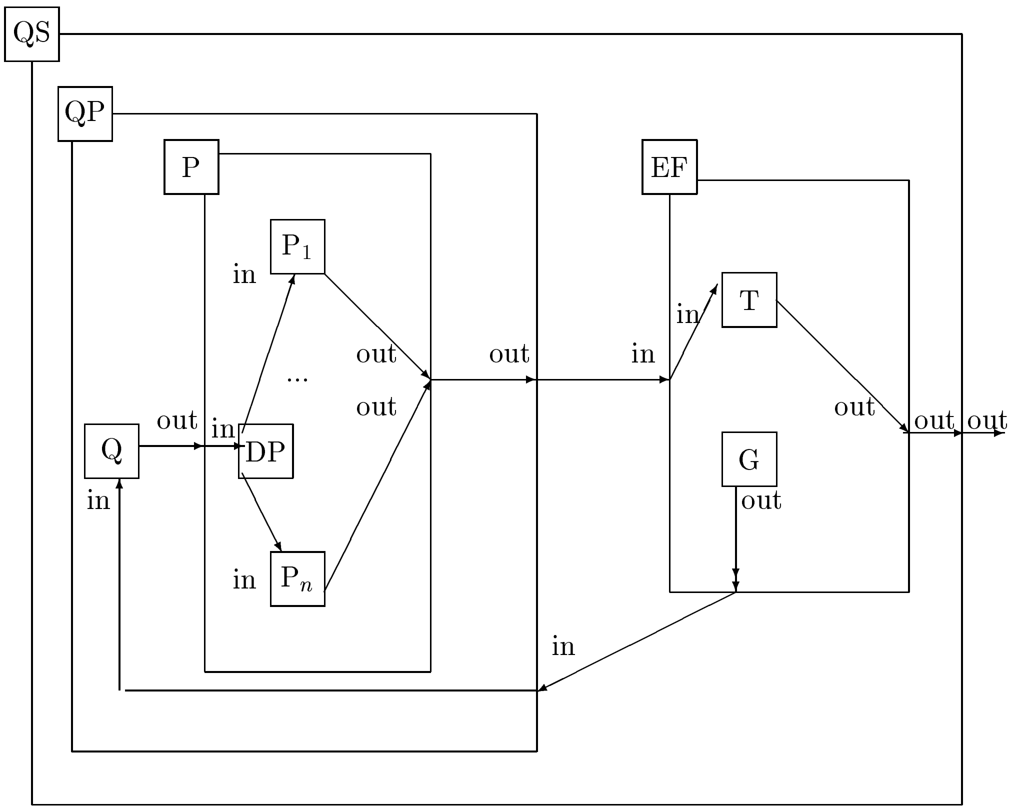

Figure 1: The static structure of the models

The identifiers write with capital, respectively small letters are indicating the models identifiers, respectively the ports denominations. The arrows are showing the direction of the messages between models. In the next subsection, the content of the messages will be presented.

3.2 The DEVS model description

The next theorem supplies the DEVS representation of the queue model.
Theorem 1. The queue is a DEVS model.
Proof. The set of inputs in the queue is:
$X_Q = \{(i, gtime, \sigma) \,/ i = 1, .., m; gtime, \sigma \in R_+ \cup \{0\}\} \cup \{(\nu, kpe, tce, pie) \,/$
$\nu \in R_+ \cup \{0\}, kpe = 0, .., m; tce \in \{-1, 0, 1\}; pie, \nu \in R_+ \cup \{0\}\}$.

An $(i, gtime, \sigma)$ input means the arrival of a i class client, at the moment $gtime$, σ being the service time of this customer. For the second mesage type, ν is the time interval until a working station becomes free, i.e. a client can be served; $tce = -1, kpe = 0, ipe = 0$ means a preemption had not take place, else if $tce \in \{0, 1\}, kpe = 1, .., m$ and $ipe > 0$; tce indicates the type of the preempted client; kpe, contains the class of the preempted customer, and ipe, is the remaining time until the service completion.

A state s_Q is a list of form $s_Q = (qtime, q_1, .., q_m, \theta, ie, kev, tcev)$; $qtime$ is the event time of the last change of the model state; initialy qtime is zero; q_i is Λ, if there are not customers of i class in the queue, or a list of the form $q_i = \left(\left(tc_1^i, \tau_1^i, \sigma_1^i\right), .., \left(tc_{n_i}^i \tau_{n_i}^i, \sigma_{n_i}^i\right)\right)$ i.e. there are n_i customers in the queue, $n_i \geq 1$, tc_i^j, respectively τ_i^j and σ_i^j contain the type, respectively the waiting time in the queue, since the moment of the arrival and the service time of the j-th client of the i class, $(i = 1, .., m; j = 1, .., n_i)$; θ is the time interval after that a working station becomes free; *ie=0, kev=0* and *tcev=-1*, if this state is not the preemption result, otherwise *ie, kev*, respectively *tcev* are the time interval remaining until the service completion, the class, respectively the type of the last preempted customer.

Let k given by the following relations

$$k = 0, \ if \ q_j = \Lambda, \ for \ all \ j, \ j = 1, ., m \tag{3}$$

$$k = min\{j/q_j \neq \Lambda, j = 1, .., m\}, \ if \ there \ is \ j, \ so \ that \ q_j \neq \Lambda, \ j = 1, .., m \tag{4}$$

The advanced time function is: $ta_Q(s_Q) = \infty, if \ k = 0; \ \theta, \ in \ other \ case.$ The internal transition function is expressed as follows; be $s_Q^{'} = \delta_{intQ}(s_Q) =$
$= \left(qtime^{'}, q_1^{'}, .., q_m^{'}, \theta^{'}, kev^{'}, tcev^{'}\right)$, then $qtime^{'} = qtime + ta_Q(s_Q)$, $\theta^{'} = \infty$, $kev^{'} = 0, tcev^{'} = -1$ and any $q_j^{'}\,(j = 1, .., m)$ is given by one of the following relations:

$$q_j^{'} = \left(\left(tc_1^j, \tau_1^j + \theta, \sigma_1^j\right), .., \left(tc_{n_j}^j, \tau_{n_j}^j + \theta, \sigma_{n_j}^j\right)\right), if \ j \neq k \ and \ q_j \neq \Lambda \tag{5}$$

$$q_j^{'} = \left(\left(tc_2^j, \tau_2^j + \theta, \sigma_2^j\right), .., \left(tc_{n_j}^j \tau_{n_j}^j + \theta, \sigma_{n_j}^j\right)\right), \ ifj = k \ and \ n_j \geq 2 \tag{6}$$

$$q_j^{'} = \Lambda, \ if \ j = k \ and \ n_j = 1 \tag{7}$$

$$q_j^{'} = \Lambda, \ if \ q_j = \Lambda \ and \ j \neq k \tag{8}$$

The outputs of the model are $\left(k, \tau_1^k + \theta, tc_1^k, ie, kev, tcev\right)$, sended to EF model and $\left(k, qtime, tc_1^k, \sigma_1^k\right)$, sended to P model.

If the input is $(i, gtime, \sigma)$, let $s_Q^{'} = \delta_{extQ}\left(s_Q, (i, gtime, \sigma), e\right) = \left(qtime^{'}, q_1^{'}, .., q_m^{'}, \theta'\right)$; then: $qtime^{'} = min\,(qtime, gtime)$,

$$\theta^{'} = 0,\ if\ qtime \leq gtime; \tag{9}$$

$$\theta^{'} = qtime - gtime,\ if\ qtime > gtime \tag{10}$$

$q_j^{'}, (j = 1, .., m)$ is given by:

$$q_j^{'} = \left(\left(tc_1^j, \tau_1^j, \sigma_1^j\right), .., \left(tc_{n_j}^j, \tau_{n_j}^j, \sigma_{n_j}^j\right)\right),\ if\ j \neq i\ and\ q_j \neq \Lambda \tag{11}$$

$$q_j^{'} = \left(\left(tc_1^j, \tau_1^j, \sigma_1^j\right), .., \left(tc_{n_j}^j, \tau_{n_j}^j, \sigma_{n_j}^j\right), (0, 0, \sigma)\right),\ if\ j = i\ and\ q_j = \Lambda \tag{12}$$

$$q_j^{'} = \Lambda,\ if j \neq i\ and\ q_j = \Lambda \tag{13}$$

$$q_j^{'} = ((0, 0, \sigma))\ \ if\ j = i\ and\ q_i = \Lambda \tag{14}$$

Once with this external transition, an internal transition takes place, which not changes the internal state ofthe model, but generates an output of the form *(i,qtime,"new-arriv")*. If the received message is (ν, kpe, tce, pie), then the external transitionis described by: $ie^{'} = pie; tcev^{'} = tce, \theta^{'} = \nu; kev^{'} = kpe$ and any $q_j^{'}, (j = 1, .., m)$, $j = 1, .., m$, is given by:

$$q_j^{'} = \left(\left(tc_1^j, \tau_1^j - ie^{'}, \sigma_1^j\right), .., \left(tc_{n_j}^j, \tau_{n_j}^j - ie^{'}, \sigma_{n_j}^j\right)\right),\ if\ j \neq k\ and\ q_j \neq \Lambda \tag{15}$$

$$q_j^{'} = \left(\left(1, 0, ie^{'}\right), \left(tc_1^j, \tau_1^j - ie^{'}, \sigma_1^j\right), .., \left(tc_{n_j}^j, \tau_{n_j}^j - ie^{'}, \sigma_{n_j}^j\right), \right),$$

$$if\ j = kpe\ and\ q_j = \Lambda \tag{16}$$

$$q_j^{'} = \Lambda,\ if j \neq kpe\ and\ q_j = \Lambda \tag{17}$$

$$q_j^{'} = \left(\left(1, 0, ie^{'}\right)\right)\ \ if\ j = kpe\ and\ q_j = \Lambda \tag{18}$$

The time interval e since the last internal transition is given by:

$$e = qtime - gtime,\ if\ qtime > gtime \tag{19}$$

$$e = gtime - qtime,\ if\ qtime \leq gtime \tag{20}$$

Remark 1. If k is given by the (3) relation, there are not clients in the queue, else k, given by the (4) relation is the index of the class that will give the customer to be served. If at a moment t, there are not clients in the queue and the service stations are free, the state of the model is passive and an arrival activates him. This is the model initial state too.

Remark 2. The (5) relation means that in the not empty queue, from wich the customer is no selected, only the waiting time component is modified, being incremented with the value of the time advanced function; the (6) relation says that from the minimum index queue on extract the first client in order to be served and the waiting times of the left customers are modified as in the

precedent case; the (7) relation means that if the minimum index queue have a single element, it becomes empty. The (8) relation says that the empty queue left empty, after this transition.
Remark 3. The (11) relation says that the not empty queue, of the index not equal with the arrival index not changes; the (12) and (14) relations say that the new client arrived, joins at the end of his class, the initial waiting time and client type being 0; the (13) relation says that the empty class different of the arrived client class remain empty.
Remark 4. The (15) and (16) relations say that the remaining time interval until the service completion of the last preempted client, is substracted from the total waiting time in the queue of the clients, because it previously was added, once the internal transition; also, the (15) and (17) relations says that the preempted client is firstly joined at his class.
Remark 5. The *qtime* $\geq$ *gtime* condition from (9) relation, is equivalent with $ta_Q(s_Q) < \infty$, i.e. there are clients in the system; the complementary condition says that we have a passive model and the next possible event will be one arrival, after a *gtime-qtime* time interval, since the last transition.

Any active working station is a model and because their number is variable, it is natural to consider the model obtained by coupling the active working stations, denoted by P as being one with a variable structure. So, at a given moment if $D^{DP} = \{P_1, ..., P_r\} \subseteq \{1, .., n\}$ is the set of active working stations, the variable structure network model is $DSDEVN_P = (DP, M_{DP})$, where $M_{DP} = (X_{DP}, Y_{DP}, S_{DP}, \delta_{int_{DP}}, \delta_{ext_{DP}}, ta_{DP}, \lambda_{DP})$ is the DEVS model of the network executive.

The DEVS representation of the serving station model is given by the following theorem.
Theorem 2. Any active working station is a DEVS model.
Proof. Let $M_j = \left(X_j, Y_j, S_j, \delta_{int_j}, \delta_{ext_j}, ta_j, \lambda_j\right), j \in D^P$ a DEVS model of active working station *j*. An input of this model is of form $(i, tc, qtime, \sigma)$. If the station is active, then his state is of the form $s_j = (tc_j, k_j, ptime_j, \sigma_j)$, where $tc_j, i_j, ptime_j, \sigma_j$ have the previous significance. If the station has been finished to serve the client, then his state is $(-1, 0, \infty, 0)$. When it receives an external message, then an external transition is carried out; if $s_j^{'} = \left(tc_j^{'}, i_j^{'}, ptime_j^{'}, \sigma_j^{'}\right)$ is the new state, then $tc_j^{'} = tc, i_j^{'} = i, ptime_j^{'} = qtime + \sigma, \sigma_j^{'} = \sigma$. After this one, an internal transition will follow, the new state being $(-1, 0, \infty, 0)$ and $ta_j(s_j) = \sigma_j$. The model output is $(j, tc_j, i_j, \sigma_j, ptime_j)$.

The next theorem proves the closure of the P model, versus the coupled relation.
Theorem 3. The variable structure network model P is a DEVS model.
Proof. Let $M_P = (X_P, Y_P, S_P, \delta_{int_P}, \delta_{ext_P}, ta_P, \lambda_P)$ the DEVS model of P. A state of this model is of the form $s_P = ((s_{DP}, e_{DP}), (s_{P_1}, e_{P_1}), ..., (s_{P_r}, e_{P_r}))$.
$s_{DP} = \left(X_P^{DP}, Y_P^{DP}, D^{DP}, \{M_i^{DP}\}, \{I_i^{DP}\}, \{Z_{i,j}^{DP}\}, sel^{DP}, ql, ptime, pf\right)$ where:
$X_P^{DP} = \{(tc, i, qtime, \sigma, ie) / tc \in \{0,1\}, i = 1, .., m; ie, \sigma, qtime \in R_+ \cup \{0\}\} \cup \{i, gtime, \text{"}newarriv\text{"} / i = 1, .., m, gtime \in R_+ \cup \{0\}\}$ is the inputs set of the network; the first input message means that one client of *i*-class will be served, *tc*,

respectively σ being the client type, respectively the time interval of the service and *qtime* is the moment when this action takes place; the second indicates that a new client joins at the *i*-class, at the moment *qtime*. Y_P^{DP} is the outputs set of the network,
$Y_P^{DP} = \{(\nu, kpe, tce, pie) / \nu, pie \in R_+ \cup \{0, \infty\}, kpe = 0, .., m; tce \in \{-1, 0, 1\}\}$
$\cup \{(tc, ptime, j, i, \sigma, tlen) / j = 1, .., n; i = 1, .., m, \sigma, ptime, tlen \in R_+\}$
$\cup \{(pe, tce, ke, ipe) / tce \in \{0, 1\}, pe = 1, .., n; ke = 1, .., m; ipe \in R_+ \cup \{0\}\}$;
ν is the time interval after that the first active station will finish his service, i.e. a new client may be served; $\nu = 0, tce = -1; kpe = 0, pie = \infty$ says that a preemption will not takes place, otherwise $(kpe > 0, tce \geq 0, pie < \infty)$, the preemption will take place, *kpe*, respectively *tce*, and *pie* being the class, respectively the type and the remaining time until the service completion of the preempted client; $(tc, ptime, j, i, \sigma, tlen)$ means that the *j*-station has served a client from *i*-class, of *tc*- type, *ptime* is the moment of the serving finish and σ the length of the service; *(pe,tce,ke,ipe)* means that the *pe*-station was preempted, *tce*, respectively *ke* is the type, respectively the class of the preempted client and *ipe* is the remaining time until the service completion. The significance of D^{DP} was above presented. $M_j^{DP}, j \in D^{DP}$ is the DEVS model of the *j*-active working station. $I_{DP}^{DP} = \{P, P_1, ..., P_r\}$, $I_j = \{P\}, j \in \{P_1, .., P_r\}$, $I_P = \{DP\}$. In all cases, $Z_{i,j}$ is the identical function. *ql* is the length of the queue. *pf=0*, if all active working stations are serving the clients or $pf \in \{1, .., n\}$, if one active station has finished his service and it is waiting in order to serve a new client. *ptime* is the moment of the next service finishing.
External transition. When receives an input of the form $(tc, i, qtime, \sigma)$, this is sent to the executive, his effect being: *ql:=ql-1*; the message is sent to the *pf*-station that will carry out himself external transition and *pf* will become 0; also, $Tlen := Tlen + qtime - ptime,\ if\ D^{DP} = \{\}, pf \in \{1, 2, .., n\}\ and\ s_{pf} = (0, 0, \infty)$. When the input is *(i,qtime, "new-arriv")*, this message is sent to the executive; the next actions take place: *ql:=ql+1*; if $ql/nbp \geq lmax\ and\ nbp < n$ then the station $P_{r+1} = min\{j/j\ not\ in\ D^{DP}\}$ will become active; $pf = P_{r+1}$; $D^{DP} = \{P_1, ..., P_r, P_{r+1}\}$; an instantaneous internal transition is carried out that not changes the model stat, but a (ν, kpe, tce, pie) output message, with $\nu = 0, kpe = 0, tce = -1, pie = \infty$, is generated; otherwise, be $km = min\{i_j/j \in D^{DP}\}$; there are two cases:
- if $i \geq km$, then the preemption will not take place and a (ν, kpe, tce, pie) message, with $\nu = \infty, kpe = 0, tce = -1, pie = \infty$ is generated;
- $i < km$, then the preemption will take place; be Δ, l given by:
$\Delta = min\{ptime_j - qtime/i_j = km, j \in D^{DP}\}$,
$l = min\{j/ptime_j - qtime = \Delta, i_j = km, j \in D^{DP}\}$.
The state of model will change as follows: *pf=l, ptime=qtime*; the messages (ν, tce, kpe, pie), with $\nu = 0, pie = \Delta, tce = tc_l, kpe = i_l$ and (pe, tce, ke, ipe), with $pe = l, tce = tc_l, ke = i_l, ipe = \Delta$ will be sended to Q, respectively to EF model.
Internal transition. Be $\Delta = min\{ta_i(s_i) - e_i/i \in D^{DP}\}$,
$Im = \{r/r \in D^{DP}, ta_r(s_r) - e_r = \Delta\}$; $i^* = sel^{DP}(Im) = min\{r/r \in Im\}$; i^* will generate the next internal transition. The internal transition takes place

in two steps; firstly, the efect about the i^* component is established and the second is the effect about the influenced components. So, after the first step, the model state will become:
$s_P = ((s_{DP}, e_{DP} + ta_{i^*}(s_{i*})), (s_{P_1}, e_{P_1} + ta_{i^*}(s_{i*})), .., ((\delta_{int_{i^*}}(s_{i*}), 0), ..,$
$(s_{P_r}, e_{P_r} + ta_{i^*}(s_{i*})) = ((s_{DP}, e_{DP} + ta_{i^*}(s_{i*})), (s_{P_1}, e_{P_1} + ta_{i^*}(s_{i*})), ..,$
$\left(s_{pred(i^*)}, e_{pred(i^*)} + ta_{i^*}(s_{i*})\right), \left(s_{succ(i^*)}, e_{succ(i^*)} + ta_{i^*}(s_{i*})\right), ...,$
$(s_{P_r}, e_{P_r} + ta_{i^*}(s_{i*}))$.
After the second step, only the state of the executive will be change, i.e.:
- $ptime := ptime_{i^*}$;
- if $ql/nbp \geq lmin$, then $pf = i^*$ and the output generated will be $(\nu, -1, 0, \infty)$, with $\nu = \Delta$;
- if $ql/nbp < lmin$ *and* $nbp > 1$ then $D^{DP} := D^{DP} - \{i^*\}$ and the output generated will be $(\nu, -1, 0, \infty)$, with $\nu = \infty$;
- if $ql/nbp < lmin$ *and* $nbp = 1$ then $pf = i^*$, $D^{DP} := \{\}$ and the output generated will be $(\nu, -1, 0, \infty)$, with $\nu = \Delta$.
In all cases the output $(tc_{i^*}, ptime, i^*, i_{i^*}, \sigma_{i^*}, Tlen)$ is generated.

The models P and Q can be coupled, and we obtain the fixed network structure model QP. In the following, we show that this one is itself a DEVS model.
Theorem 4. We can describe QP as a DEVS model.
Proof. The model inputs are the same as the inputs of the queue. Because only the queue is connected to exterior, when QP receives an external message, this one will be send to the queue, that will perform his external transition. The state of the M_{QP} model is of form $s_{QP} = ((s_Q, e_P), (s_P, e_P))$. According to the "next event" or "minimum time" rule, the change of the coupled model will be generate by the component for that on reaches the value θ, given by $\theta = min\{ta_Q(s_Q) - e_Q, ta_P(s_P) - e_P\}$. Let $s'_{QP} = \left(\left(s'_Q, e'_Q\right), \left(s'_P, e'_P\right)\right)$ the new state obtained because of the internal transition. If $\theta = ta_Q(s_Q) - e_Q = ta_P(s_P) - e_P$ then the sel_{QP} function will supply the Q component, which firstly will perform their internal transition. So, if $ta_Q(s_Q) - e_Q \leq \mathrm{ta}_P(s_P) - e_P$, then the Q model will generate the transition of the coupled model and the new state is $s'_Q = \delta_{int_Q}(s_Q), e'_Q = 0, s'_P = \delta_{ext_P}(s_P, (tc, i, qtime, \sigma), e), e'_P = 0$. If $ta_Q(s_Q) - e_Q > ta_P(s_P) - e_P$, i.e. the next component that will perform the internal transition is the P model, then $s'_Q = \delta'_{ext}(s_Q, (\nu, tce, kpe, ie), e)), e'_P = 0, s'_P = \delta_{int_P}(s_P), e'_P = 0$. The model outputs are the Q or P model wich are above presented and that are sended to the EF model.

The experimental frame is a fixed network model, obtained by coupling the transducer and the generator models. The generator model implements the arrivals mechanism and the transducer model updates certain variables, based one the system efficiency factors are computed. In the following, the DEVS representation of these models is presented.
Theorem 5. The generator is an DEVS model.
Proof. The set of the inputs is a empty set, so we have an autonomous model. His state is of form $s_G = (i, gtime, \sigma)$, where i is the class of the generated job, $gtime$ is the value of the generator clock and σ is the serving time interval value of this client. The output of this model is of form $(i, gtime, \sigma)$. Let AT the random variable of the inter-arrival time, T the random variable

of the priority class and ST the random variable of the service time. Because we have an autonomous model, only the internal transition will take place. If $s_G = (i, gtime, \sigma)$ and $s_G^{'} = \delta_{int_G}(s_G)$, then $s_G^{'} = \left(i^{'}, gtime^{'}, \sigma^{'}\right)$, where $i^{'} = T$, $gtime^{'} = gtime + AT$ and $\sigma^{'} = ST$; initially *gtime=0.*

Theorem 6. The transducer is a DEVS model.

Proof. The set of inputs is:

$X_T = \{(tc, k, \tau + \theta, ie, ke, tce) / tc, tce \in \{-1, 0, 1\}; ke, k = 0, 1, .., m; \tau, ie \in R_+ \cup \{0\}\} \cup$

$\cup \{(tc, i, k, \sigma, ptime, tlen) / tc \in \{0, 1\}; i = 1, .., n; k = 1, .., m; \sigma, ptime, tlen \in R_+ \cup \{0\}\} \cup$

$\cup \{(tc, kp, pe, ipe) / tc \in \{0, 1\}; i = 1, .., n; k = 1, .., m; \sigma, ptime \in R_+ \cup \{0, 1\}\}.$

A state of this model is of form $s_T = (Tt, Ttl, Tst, Tste, Twt, Twte, Ns, Nse)$, where: Tt is the value of the clock; Ttl is the total laziness time; Tst and Tst are bi-dimensional vectors, *Tst(i,k)*, respectively *Tste(i,k) (i=1,..,n; k=1,..,m)*, means the total working time of i station in order to serve the not preempted, respectively preempted clients of k class; Twt and $Twte$ are vectors, *Twt(i)*, respectively *Twte(i)(i=1,..,m)* is the total waiting time in the queue of the not-preempted, respectively preempted i-class clients; *Ns* and *Nse* are bi-dimensional vectors, *Ns(i,k)*, respectively *Nse(i,k) (i=1,..,n; k=1,..,k)* contains the total numbers of the not preempted, respectively preempted customers of k-class, served by the i-station. This model is passive, so only external transitions are performed.

When it receives an input of form $(tc, i, \tau + \theta, ie, ke, tce)$ the model performs an external transition, as follows: if *tc=0*, then $Twt(i) := Twt(i) + \tau$, else $Twte(i) := Twte(i) + \tau$. Besides, if $ie > 0$ then: if *tce=0*, then *Twt(ke):=Twt(ke)-ie* else, *Twte(ke):=Twte(ke)-ie.*

When it receives an input of form *(tc,kp,pe,ipe)* then,
if *tc=0* then *Tst(pe,kp):=Tst(pe,kp)-ipe*, else *Tste(pe,kp):=Tste(pe,kp)-ipe.*

When it receives an input of form $(tc, i, k, \sigma, ptime, Tlen)$ then: *Tt:=ptime;*
if *tc=0* then $Tst(i, k) := Tst(i, k) + \sigma$, *Ns(i,k):= Ns(i,k)+1,*
else $Tste(i, k) := Tste(i, k) + \sigma$, *Nse(i,k):= Nse(i,k)+1*; *Ttl:=Ttl+tlen.*

Because the model is passive, then $\mathrm{ta}_T(s_T) = \infty$.

Remark 4. Based on the model state variables, the model efficiency factors are computed, as follows:

- The average of the waiting time in the queue of the k- class not preempted clients, $MTw(k) = \frac{Twt(k)}{Ns(k)}, k = 1, .., m.$
- The average of the waiting time in the queue of the k-class preempted clients, $MTwe(k) = \frac{Twte(k)}{Nse(k)}, k = 1, .., m.$
- The average of the service time of the k-class not preempted clients, $MTs(k) = \frac{\sum_{i=1}^{n} Tst(i,k)}{\sum_{i=1}^{n} Ns(i,k)}, k = 1, .., m.$
- The average of the service time of the k-class preempted clients, $MTse(k) = \frac{\sum_{i=1}^{n} Tste(i,k)}{\sum_{i=1}^{n} Nse(i,k)}, k = 1, .., m.$
- The average of the number of k-class not preempted clients from the queue, $MLQ(k) = \frac{Twt(k)}{Tt}, k = 1, .., m;$
- The average of the number of k-class preempted clients from the queue,

$MLQe\left(k\right)=\frac{Twte(k)}{Tt}, k=1,..,m;$
- The average of the length of the queue, $MQueue=\sum_{i=1}^{m}\left(MLQ\left(i\right)+MLQe\left(i\right)\right).$
- The average of the stations laziness time, $MTtl=\frac{Ttl}{n}$ and the laziness coefficient $MTtl/Tt$.
- The average of the service time of the i station, $MTts\left(i\right)=\frac{\sum_{k=1}^{m}(Tst(i,k)+Tste(i,k))}{\sum_{k=1}^{m}(Ns(i,k)+Nse(i,k))}.$

Remark 5. The closure under coupling of the QS and EF models, may be proved in a similar mod as in the QP coupled model case.

4 The abstract simulator associated to the DEVS models

The abstract simulator [17, 18, 19] is an algorithmic description of the behavior of the corresponding model.Corresponding to the hierarchical construction of the DEVS model, there is that of the abstract simulators. At the high level in this construction, is the root coordinator that initiates the simulation. In figure 2, the hierarchical construction of the DEVS abstract simulator is illustrated. At the level of atomic models there is the simulator class. At the level of the coupled model with fixed structure, respectively variable structure there are the coordinator class, respectively network simulator class. The simulator implements the internal transition, the external transition and the time advance function of the atomic models. The coordinator and the network simulators make the synchronization between the subordinated abstract simulators.

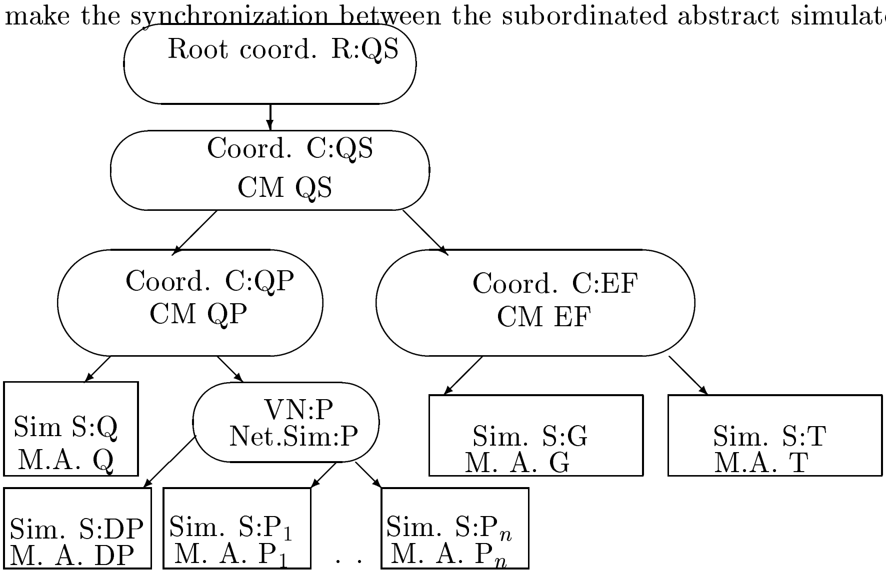

Figure 2: The abstract simulator structure and hierarchical construction (M.A.=Atomic Model, C.M=Coupled Model with fixed structure, V.N.M=Variable structure Network Model)

A very important aspect of the simulation of the DEVS models, is the communication between the abstract simulators. The sending of the messages between them makes this. Indifferently of the type of the abstract simulator, there is five message types; an i-message corresponds to establish an initial state; a *-message, respectively a x-message, received from a coordinator at the immediately superior level, means the performing of an internal transition, respectively an external transition. These messages are going from top to bottom. A y-message, respectively a *done*- message is going in the reverse way; the first contains the output message generated because an internal transition takes place, respectively the second contains the acknowledgement of the external transition execution.

The root coordinator tries the simulation by an i-message sending to the high level coordinator, that makes the same thing. Every abstract simulator confirms his initialisation by sending a *done*-message to the coordinator at the immediately level. After the coordinator from the high level has received the acknowledgement message from their subordinates, the sending of one number of *-messages follows, until a finishing simulation criterion is accomplished. This criterion may be the number of the simulation events or to reach by the global simulation clock a maximum given value.

When a coordinator receives a *-message, it send this message to his sun with the minimum time of the next event (*tn*). If this is a simulator, then it perform his internal transition and responds with an y-message to his parent coordinator, otherwise the message is sent down to the subordinated abstract simulator with minimum *tn*, and so on and so forth. On an internal transition are sent messages to the influenced simulators, based on these models external transitions will perform.When a coordinator receives a x-message, it checks if the event time is greater than the time of the last event (*tl*) and the time of the next event and this message is divided into other messages that are sent to the connected to exterior subordinates, that perform himself the external transition and confirm by done-messages. When a simulator receives a x- message it performs the external transition, and confirm by a *done*-message. In all transitions, the *tl* and *tn* variables are updating. In the following, the DEVS abstract simulators are presented.

The simulator S:Q of the DEVS Q model

When receives (i,t)

$tl_Q := t; tn_Q := \infty;$*done:=true*; *send (done,t) to C:QP.*

When receives (,t)*

done:=false;

$If\ t \neq\ tn_Q\ then\ 'Error - Bad\ synchronisation'$

else

$k :=$ *1*;

While $(k \leq\ m\ )$ *and* $(q_k\ \neq\ \Lambda)$ *do* $k := k+1;$

done:=false;

if $k \leq m$ *then*

$tn_Q := tn_Q + \theta;$

$send\Big(\left(k, tc_1^k, \tau_1^k + \theta, ie, kev, tcev\right), \left(tc_1^k, tn_Q, \sigma_1^k\right), t\Big)$ *to C:QP*;

$s_Q := \delta_{int_Q}(s_Q)$;{*get off the pair* $\left(tc_1^k, \sigma_1^k, \tau_1^k\right)$ *from the queue* }

$tl_Q := t;\ done := true;$

else "Error:";

send (done,t) to C:Q .
When receives ((i,gtime,σ), t);
done:=false;
$if\ not(tl_Q \leq t \leq tn_Q)$ *then* $'Error - Bad\ synchronisation'$
else
$s_Q := \delta_{ext_Q}(s_Q, t - tl_Q, (i, gtime, \sigma))$; $\{(0, 0, \sigma)$ *get on in the queue*$\}$
done:=true; send (done,(i,tn$_Q$,"new arriv"),t) to C:QP.
When receives ((ν, kpe, tce, pie),t)
$\theta := \nu$; *done:=true;* $s'_Q = \delta_{ext_Q}(s_Q)$;$\{$ $(1, kpe, pie)$ *joins to the queue* $\}$
send (done) to C:QP

The abstract simulator of the variable structure DEVS model P is a network simulator, NS:P. Firstly, the simulator associated of every active working station j, $j \in \{1, .., n\}$ is presented.
When receives (i,t) ;
$tl_j := t; tn_j := \infty; done := true;$
send (done,t) to NS:P.
When receive$((k, tc, tn^Q, \sigma), t)$
done:=false;
$if\ not(tl_j \leq t \leq tn_j)$ *then* $'Error - Bad\ synchronisation'$
else
$s_j := \delta_{ext}(s_j, t - tl_j, (k, tc, tn^Q, \sigma)); \{s_j = (tc, k, tn^Q, \sigma)\}$
$tl_j := t; tn_j := tl_j + \sigma; done := true;$
send (done) to NS:P;
*When receives (*,t)*
done:=false;
$If\ t \neq tn_j$ *then*$'Error - Bad\ synchronisation'$
Else
$tl_j := t; tn_j := \infty; ptime_j := ptime_j + \sigma;$
$s_j := \delta_{int_j}(s_j); done := true\ \{s_j = (-1, 0, \infty, 0)\}$
$send((j, ptime_j, tc_j, k_j, \sigma_j), t)\ to\ NS : P;$
send (done) to NS:P.

The network simulator NS:P of model P
When receives (i,t) ;
$D^{DP} := \{1\}$*;send(i,t) to S:1;*
Wait Until(done) from S:1;
$tl_P := t; tn_P := \infty$*; ql:=0; nbp:=0;pf:=1;TTl:=0;read(Lmax,Lmin);*
When receives ((k,tc,tn$_Q$,σ),t)
$if\ t \notin [tl_P, tn_P]$ *then* $ERROR$; *end.*
send ((k,tc,tn$_Q$,σ),t) to S:pf;
WaitUntil (done,t) from S:pf;
$tl_P := t; tn_P := min\{tn_i / i \in D^{DP}\};$
send(done,t) to C:QP;
When receives ((i,tn$_Q$,"new arriv"),t) ;
$tl_P := t; tn_P := t; ql := ql + 1;$
if $(ql/nbp > Lmax)$ *and* $(nbp < n)$ *then*

$\nu := 0; P_{nbp+1} := min\{j/j = 1,..,n; j \; is \; not \; active\}$;
$D^{DP} := D^{DP} \cup \{P_{nbp+1}\}; pf := P_{nbp+1}$;
send(i,t) to pf;
Wait Until done from S:pf;
else
$km = min\{i_j/j \in D^{DP}\}$;
if $i \geq km$ *then* {*the preemption will not take place* }
$\nu = \infty; kpe = 0; tce = -1; pie = \infty$;
send ((ν, kpe, tce, pie)*,t)*
else {*then the preemption will take place* };
$\Delta := min\{ptime_j - tn_Q/i_j = km, j \in D^{DP}\}$,
$l := min\{j/ptime_j - tn_Q = \Delta, i_j = km, j \in D^{DP}\}$;
pf=l; ptime=tn_Q*;* $\nu = 0; pie = \Delta; tce = tc_l; kpe = i_l$;
$pe = l; tce = tc_l; ke = i_l; ipe = \Delta$*; done:=true;*
send(done,(ν, tce, kpe, pie)*,* (pe, tce, ke, ipe)*,t) to C:QP*
*When receives (*,t)*
$i^* := min\{i/tn_i = tn_P; i \in D^{DP}\}$; *Send (*,t) to S:* i^*;
WaitUntil(done,$(i^*, k_{i^*}, \sigma_{i^*}, tn_{i^*})$*) from* i^*;
tl_P*:=t;*tn_P*:=*$tn_i{}^*$;
$send(done, (i^*, k_{i^*}, \sigma_{i^*}, ptime_i{}^*)) to C : QP$;
if $(ql/nbp < Lmin)$ *then*
if $(nbp > 1)$ *then*
$D^{DP} := D^{DP} - \{i^*\}; nbp := nbp - 1; \nu := \infty$;
$kpe = 0; tce = -1; pie = \infty$;
else $pf := i^*; \nu := \Delta; kpe = 0; tce = -1; pie = \infty$;
else $pf := i^*; \nu := \Delta; kpe = 0; tce = -1; pie = \infty$;
send(done,(ν, tce, kpe, pie) *,t) to C:QP;*
end.

The abstract simulator associated to **the experimental frame is a coordinator (C:EF)**. It make the synchronization of the simulators associated to the generator (S:G) and transducer (S:T) models. The **S:G simulator** is:
When receives (i, t)
Read(The parameters of random generated variables T,AT, ST);
tl_G*:=t;* tn_G*:=t;done:=true;*
send (done,t) to C:EF.
*when receive (*,t)*
if $t \neq tn_G$ *then "Eroare"*
else
$Gen(k); Gen(IntAriv); Gen(\sigma);_G := t$;
$tn_G := tn_G + IntAriv; s_G := \delta_{int_G}(s_G)$*; send(*$k, tn_G, \sigma$*) to C:EF;*
end.

The **simulator of the transducer DEVS** model updates the values of the variables based on the system efficiency factors are computed.
when receives (i,t)
read(Nevs) {*Nevs= Total number of served clients* };
$tl_T := t; tn_T := \infty; Tst := 0_{n,m}; Tste := 0_{n,m}; Twt := 0_m; Twte := 0_m$;
$Ns := 0_m; Nse := 0_n; evs := 0; stopsim := false; Ttlen := 0$

when receives $((tc, i, k, \sigma, Tlen), t)$

done:=true; Tt:=ptime; Ttlen:=Ttlen+Tlen;$tl_T := t$*;*

if tc=0 then $Ns(k) := Ns(k) + 1$; $Tst(i,k) := Tst(i,k) + \sigma$;

else Nse(k):= Nse(k)+1; Tste(i,k):= Tste(i,k)+σ;

if evs=Nevs then stopsim:=true;

send (done,stopsim) to C:EF

when receives ((tc,kp,pe,ipe),t)

if tc=0 then Tst(pe,kp):= Tst(pe,kp)-ipe

else Tste(pe,kp):= Tste(pe,kp)-ie;

done:=true;send (done) to C:EF;

when receives ((tc,k,$\tau + \theta$*,ie,ke,tce),t);*

if tc=0 then Twt(k):=Twt(k)+ τ

else Twte(k):=Twte(k)+ τ*;*

if ie > 0 *then*

if tce > 0 *thenTwte(ke):=Twte(ke)-ie*

else Twt(ke):=Twt(ke)-ie;

done:=true;send(done) to C:EF;

When receives(,t)*

Mqueue:=0;

for k:=1 to m do

$MTw(k) := \frac{Tw(k)}{\sum_{i=1}^{n} Ns(i,k)}$;

$MTwe(k) := \frac{Twe(k)}{\sum_{i=1}^{n} Nse(i,k)}$;

MLq(k):=Twt(k)/Tt;MLqe(k):=Twte(k)/Tt;

Mqueue:=Mqueue+ MLq(k)+MLqe(k);

$MTs(k) = \frac{\sum_{j=1}^{m} Tst(j,k)}{\sum_{j=1}^{m} Ns(j,k)}$; $MTse(k) = \frac{\sum_{j=1}^{m} Tste(j,k)}{\sum_{j=1}^{m} Nse(j,k)}$;

for i:=1 to n do

$MTss(i) = \frac{\sum_{k=1}^{m} (Tst(i,k)+Tste(i,k))}{\sum_{k=1}^{m} (Ns(i,k)+Nse(i,k))}$;

MTlen:=Tlen/n;

Clen:=MTlen/Tt;

Send (done,(Mqueue,MTw,MTwe,MTs,MTse,MLq,MLqe,MTlen,Clen,MQueue) to C:EF

End.

Remark 6. The coordinators C:EF and C:QS have a similar description with those coordinators described in this section.

5 The model validation and practical considerations

5.1 Comparing the model with a variable and a constant number of working stations

In [8] I have studied the preemptive multi-servers model with a constant number of working stations. So, I consider the model with two priority class, one serving station and the interarrival times, respectively service times, are exponential negative distribution of parameter 1, respectively 2. I have obtained the next results:

- The average of the 1-class not preempted (preempted) clients waiting time in the queue = 0.19127 (0)
- The average of the serving time of the not preemted (preempted) 1-class client = 0.51410 (0)
- The average of the 2-class not preempted (preempted) clients waiting time in the queue = 0.69507 (0.69935)
- The average of the serving time of the not preemted (preempted) 2-class client = 0.50910 (0.43235)
- The average of the station serving time = 0.44914
- The using coefficient of the working station = 0.50386.
- The average of the length of the queue = 0.37264.

For the corresponding preemptive model with a variable number of working stations, considering *Lmin=1* and *Lmax=2*, I have obtained: - The average of the 1-class not preempted (preempted) clients waiting time in the queue = 1.07762 (0)
- The average of the serving time of the not preemted (preempted) 1-class client = 0.50914 (0)
- The average of the 2-class not preempted (preempted) clients waiting time in the queue = 86.61482 (25.233255)
- The average of the serving time of the not preemted (preempted) 2-class client = 0.50310 (0.43235)
- The average of the station serving time = 0.44914
- The using coefficient of the working station = 0.50386.
- The average of the laziness coefficient of the station=0.04954
- The average of the length of the queue = 22.88242.

5.2 The study of the dependence between the *Lmax* and *Lmin* parameters and the model efficiency factors

In the following, I consider the model with three priority classes, two working stations and the inter-arrival time, respectively the serving time are values of the exponential negative random variable of parameter 0.7, respectively 1. In the figure 3 is presented the variation of the average of the wainting time in the queue, versus the values of the *Lmin*- parameter and in figure 4, the relation between the average of the laziness stations time and the *Lmax*- parameter.

5.3 Conclusions

For the system presented in (5.1), I am noting the following:
- The averages of the serving times are approximately equals;
- In the variable of number of serving stations the averages of the waiting times and the average of the length of the queue are increasing; instead, the averages of the laziness coefficients are decreasing.

In figure 3, respectively 4 we observe the decreasing of the average of the waiting time in the queue, versus the increasing of the *Lmin*-parameter, respec-

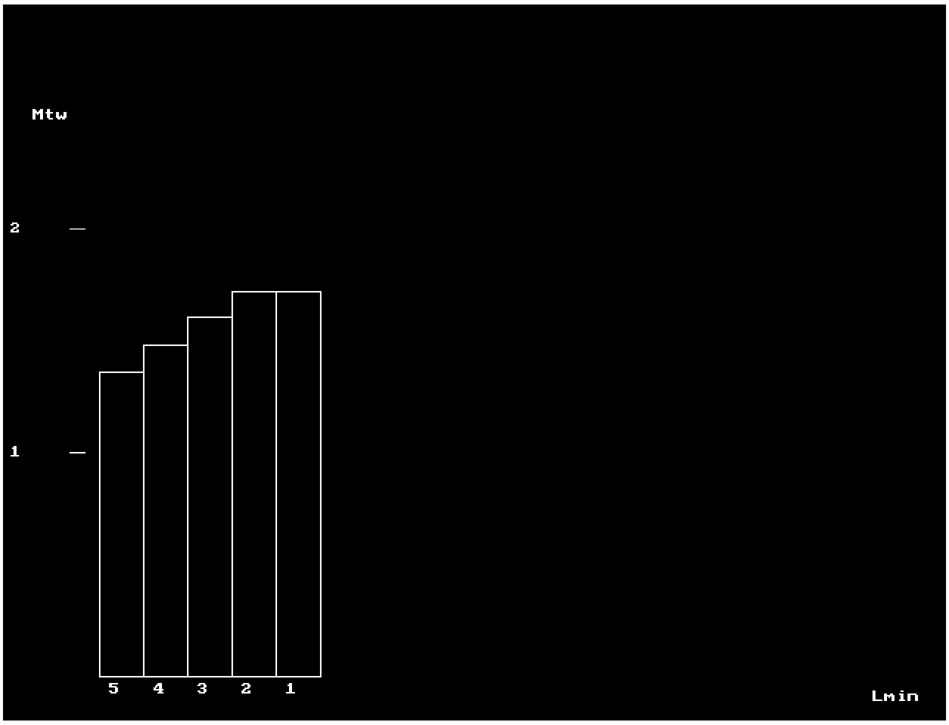

Figure 3: The relation beetwen MTw and $Lmin$

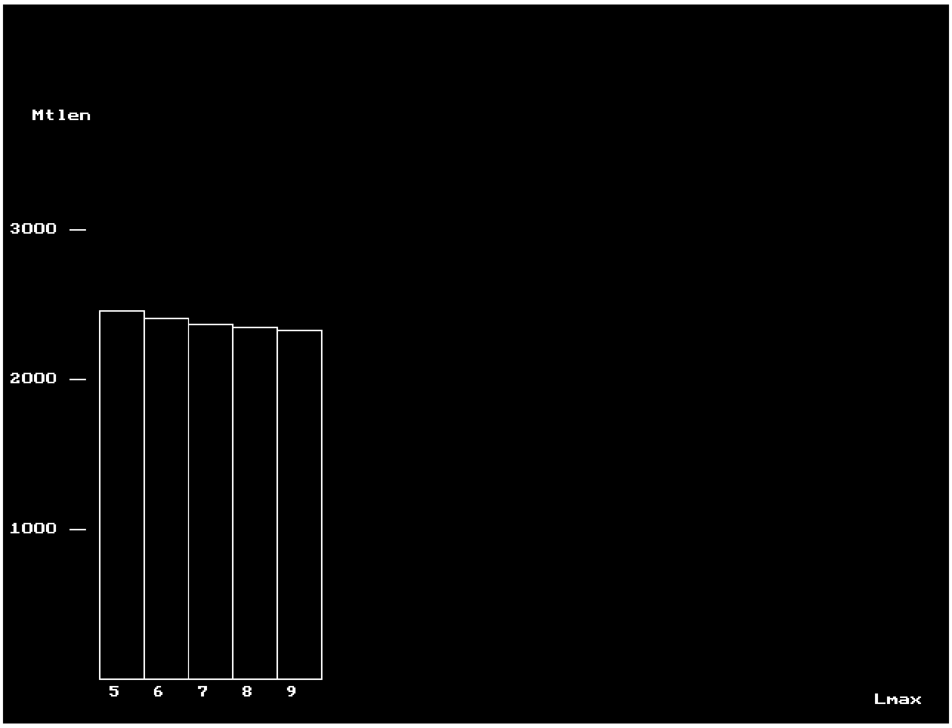

Figure 4: The dependence between $MTlen$ and $Lmax$

tively the increasing of the average of the laziness time versus the decreasing of the $Lmax$- parameter.

These results are in accordance with the assumptions based on this model.

References

[1] Barros F.J. *The Dynamic Structure Discrete Event System Specification Formalism*, Transactions of the Society for Computer Simulation International, Vol. 13, No 1, 1996.

[2] Beaumariage T., Mize J.H. *Object Oriented Modeling; Concepts and Ongoing Research*, Proceedings of the 1991 SCS Western Simulation Multiconference, La Jolla, California .

[3] Doyle R.J. *Object Oriented Simulation Programming*, Proceedings of the 1991 SCS Western Simulation Multiconference, La Jolla California, 1991

[4] Florea I. *Upon Waiting Systems of Head of Line Type: An Object Oriented Formal Approach*, Proceedings of the Balkan Conference on Operations Research, Thesaloniki, Grecia, 1997.

[5] Florea I. *One Algorithmic Approach of the Multi-Servers Head-of - Line Queuing Systems*, RAIRO - Operation Research (sent to be published), 2000

[6] Florea I. *One Algorithmic Approach of First-Come-First-Served Queuing Systems*, Annals of University of Bucharest, Informatics, 2000

[7] Florea I. *Object Oriended Simulation in the Queuing System Study* (Doctoral Dissertation), University of Bucharest, 2001

[8] Florea I. *Upon Multi- Servers Preemptive Queuing System: A Formal and Object Oriented Approach*, Annals of University of Bucharest, Informatics, 2001

[9] Kleinrock L. *Queuing Systems Volume I: Theory*, John Wiley Sons, New York, 1975

[10] Joines A. J., Roberts D. S. *Object-Oriented Simulation* (Handbook of Simulation- Banks J.), John Wiley Sons, Inc., 1998

[11] Mihoc Gh., Ciucu G., Muja A., *Waiting Mathematical Models*, Romanian Academy Publishing House, Bucharest, 1973

[12] Rozenblit J. W., *Experimental Frame Specification Methodology for Hierarchical Simulation Modeling*, International Journal of General Systems, Vol. 19, Nr. 3, 1991

[13] Sargent R.G., Mize J.H., Withers D.H., Zeigler B.P. *Hierarchical Modeling for Discrete Event Simulation*, Proceedings of the 1993 Winter Simulation Conference, Los Angeles, 1993

[14] Vaduva I. *Computer Simulation Models*, Technical Publishing House, Bucharest, 1977

[15] Vaduva I., Stoica M., Odgescu I. *Simulation of Echonomical Processes*, Technical Publishing House, Bucharest, 1983

[16] Zeigler B.P. *Theory of Modelling and Simulation*, John Wiley Sons, New York, 1976

[17] Zeigler B.P. *Multifaceted Modeling and Discrete Event Simulation*, Academic Press, 1984

[18] Zeigler B.P. *Object-Oriented Simulation with Modular Models*, Academic Press, San Diego, 1990.

[19] Zeigler B.P., Praehofer H., Kim T. G. *Theory of Modelling and Simulation. Integrating Discrete Event and Continous Complex Dynamic Systems,* Academic Press, 2000