AMO - Advanced Modeling and Optimization, Volume 3, Number 2, 2001

ONE FORMAL AND OBJECT ORIENTED APPROACH OF THE VARIABLE NUMBER OF WORKING STATIONS OF WAITING SYSTEM OF HEAD OF LINE TYPE

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 queuing systems of head - of - line type is an appropriate models for numerous real systems. In [6] I have introduced the variable number of parallel stations queuing system and in [5] I have presented simulation algorithms for the queuing system of this type, with head - of - line serving discipline. In [4], respectively in [8], the formal and object oriented models of the waiting system, with one station, respectively constant number of parallel serving stations have been proposed. Based on the DEVS formalism, that is the base of formal, modular and hierarchical modeling and object-oriented simulation of systems, in this paper I present simulation models for multi-servers queuing systems with variable number of working stations. Further, formal and modular description of systems behavior eliminates the possibility of introducing errors. To each DEVS model, an abstract simulator we associate and using of 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

The priority queuing models are very common [9, 11]. If m is the classes number, then the i-class customers 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.

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 criterion, 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.

This paper is organized as follows. Section 2 presents the framework of formal, modular and hierarchical construction of simulation models. In section 3, I present the simulation DEVS models for this queuing system class. The abstract simulators associated to simulation model are describing in section 4. The object-oriented implementation of the abstract simulator and the model validation are presenting in section 5.

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. Two relations link the basic objects: the modeling relation, which ties the real system to the model; the simulation relation, which connects model and simulator. Based on these considerations there follows the importance of finding a general framework for simulation model construction, for systems with discrete events. Such a context is the **DEVS formalism**. Thus the 7-tuple defines a model:

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

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 will reach after time e, if it has received a message x from the exterior; $\lambda : S \to Y$ is the output function, $\lambda(s)$ is the output message transmitted when the system reaches 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; communication between objects is achieved by message transmission techniques.

With regard to the modular construction of models the basic models have to be specified which underlie the construction of more complex ones, as well as the way the components are connected hierarchically. We define the **basic model** as being the model with discrete events described with the help of the DEVS formalism. Further, we define the **atomic model** as being a basic model which is not obtained by connecting other models. In the context of the DEVS formalism model connection is achieved by employing the model concept with several components, defined by the 5 -uple:

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

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. 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], the ideas are found in this paper in the construction of the DEVS model associated to the queuing system.

The experimental frame concept has been introduced in order to separate the inputs, outputs and the model dynamic [12, 17, 18]. 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, component links modifications. 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 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, the connecting relations between different models. This information may find in the **digraph model**[17, 18] that is presented in figure 1.

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 $X_Q = \{(i, gtime, \sigma)\ /i = 1, .., m;$ $gtime, \sigma \in R_+ \cup \{0\}\} \cup \{(\nu)\ /\nu \in R_+ \cup \{0\}\}$. 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; ν is the time interval after that a service station is free, so a customer

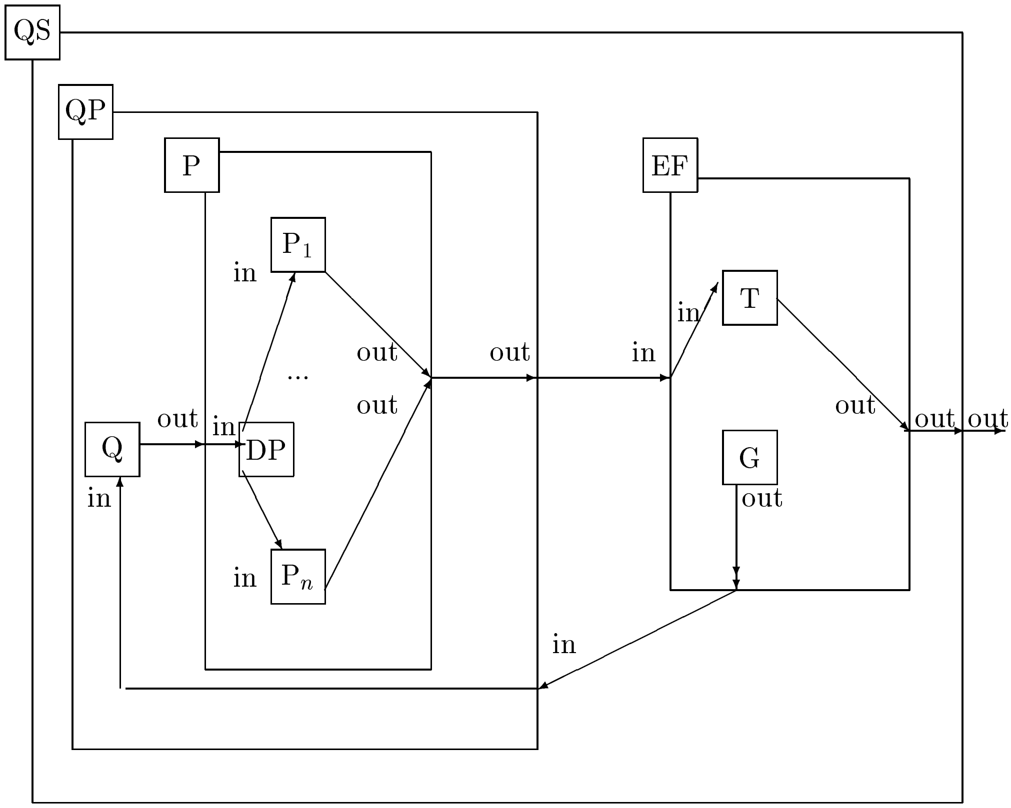

Figure 1: The static structure of the models

can be served. A state s_Q is a list of form

$$s_Q = (qtime, q_1, .., q_m, \theta) \tag{3}$$

Where *qtime* is the event time of the last change of the model state; initial qtime is zero; q_i is Λ, if there are not the customers of i class in the queue, or a list of the form

$$q_i = \left(\left(\tau_1^i, \sigma_1^i\right), .., \left(\tau_{n_i}^i, \sigma_{n_i}^i\right)\right) \tag{4}$$

i.e. there are n_i customers in the queue, $n_i \geq 1$, τ_i^j, respectively σ_i^j , being the waiting time in the queue, since the moment of the arrival, respectively the service time of the j-th client of the i class, $i = 1, .., m; j = 1, .., n_i$.
Let k given by the following relations

$$k = 0,\ if\ q_j = \Lambda,\ for\ all\ j,\ j = 1, .., m \tag{5}$$

$$k = min\{j/q_j \neq \Lambda, j = 1, .., m\},\ if\ there\ is\ j\ so\ as\ q_j \neq \Lambda,\ j = 1, .., m \tag{6}$$

The advanced time function is:

$$\mathrm{ta}_Q\left(s_Q\right) = \begin{cases} \infty, & if \quad \text{k=0;} \\ \\ \theta, & in \quad other\ case \end{cases}$$

The internal transition function is expressed as follows; if

$s_Q^{'} = \delta_{intQ}(s_Q) = \left(qtime^{'}, q_1^{'}, .., q_m^{'}, \theta^{'}\right)$, then $qtime^{'} = qtime + ta_Q(s_Q)$, $\theta^{'} = \infty$ and any queue $q_j^{'}, j = 1, .., m$ is given by the following relations:

$$q_j^{'} = \left(\left(\tau_1^j + \theta, \sigma_1^j\right), .., \left(\tau_{n_j}^j + \theta, \sigma_{n_j}^j\right)\right), if\ j \neq k,\ q_j \neq \Lambda \tag{7}$$

$$q_j^{'} = \left(\left(\tau_2^j + \theta, \sigma_2^j\right), .., \left(\tau_{n_j}^j + \theta, \sigma_{n_j}^j\right)\right),\ if j = k, n_j \geq 2 \tag{8}$$

$$q_j^{'} = \Lambda,\ if\ j = k\ and\ n_j = 1 \tag{9}$$

$$q_j^{'} = \Lambda,\ if\ q_j = \Lambda\ and\ j \neq k \tag{10}$$

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

$$\theta^{'} = 0,\ if\ qtime \leq gtime; \tag{11}$$

$$\theta^{'} = qtime - gtime,\ if\ qtime > gtime \tag{12}$$

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

$$q_j^{'} = \left(\left(\tau_1^j, \sigma_1^j\right), .., \left(\tau_{n_j}^j, \sigma_{n_j}^j\right)\right),\ if\ j \neq k\ and\ q_j \neq \Lambda \tag{13}$$

$$q_j^{'} = \left(\left(\tau_1^j, \sigma_1^j\right), .., \left(\tau_{n_j}^j, \sigma_{n_j}^j\right), (0, \sigma)\right),\ if\ j = i\ and\ q_j = \Lambda \tag{14}$$

$$q_j^{'} = \Lambda,\ if j \neq i\ and\ q_j = \Lambda \tag{15}$$

$$q_j^{'} = \Lambda,\ if\ j = i\ and\ q_i = \Lambda \tag{16}$$

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

$$e = gtime - qtime + ta_Q(s_Q),\ if\ qtime \geq gtime \tag{17}$$

$$e = gtime - qtime,\ if\ qtime < gtime \tag{18}$$

If a (ν) message is received, the θ variable become ν. The outputs of the model are $\left(k, \tau_1^k + \theta\right)$ and $\left(k, qtime, \sigma_1^k\right)$, that are sent when an internal transition takes place. When a input $(i, gtime, \sigma)$ has been received, after the external transition, an internal transition is carried out; this is an identical function that not changes the model state, but an output *'new-arriv'* is generated.

Remark 1. If k is given by the 5 relation, there are not the clients in the queue, else k is the index of the queue 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 will activate him. This is the model initial state too.

Remark 2. The 7 relation means that in the not empty queuing, from the customer is no selected, only the waiting time component is modified, being incremented with the value of the time advanced function; the 8 relation says that from the minimum index queue on extract the first client in order to be served and the waiting time of the left customers is modified as in the precedent case; the 9 relation means that if the minimum index queue have a single element, it becomes empty. The 10 relation says that the empty queue left empty, after this transition.

Remark 3. The *qtime* $\geq$ *gtime* condition from 17 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 himself and because their number is variable, it is natural to consider the model obtained by coupling the active working stations model, denoted by P as being one with a variable structure. So, if $D^{DP} = \{P_1, ..., P_r\} \subseteq \{1, .., n\}$ is the set of active working stations, at a given moment, 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 $(i, qtime, \sigma)$. If the station is active, then his state is of the form $s_j = (k_j, ptime_j, \sigma_j)$, where k_j, $ptime_j$ and σ_j have the previous significance. If the station has been finished to serve the client, then his state is $(0, \infty, 0)$. When it receives an external message, an external transition is carried out; if $s_j^{'} = \left(k_j^{'}, ptime_j^{'}, \sigma_j^{'}\right)$ is the new state, then $k_j^{'} = i, ptime_j^{'} = qtime + \sigma, \sigma_j^{'} = \sigma$. After this, an internal transition will follow, the new state being $(0, \infty, 0)$ and $ta_j(s_j) = \sigma_j$. The model output is $(j, k_j, \sigma_j, ptime_j)$.

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: $X_P^{DP} = \{(i, qtime, \sigma)/i = 1, .., m; \sigma, qtime \in R_+\} \cup \{"newarriv"\}$ is the inputs set of the network; the first input message means that one client of *i*-class will be served, σ being the time interval of the service and *qtime* is the moment when this action takes place; the second indicates that a new client has been queued. Y_P^{DP} is the outputs set of the network, $Y_P^{DP} = \{\nu/\nu \in R_+ \cup \{0\}\} \cup \{(ptime, j, k, \sigma, tlen)/j = 1, .., n; k = 1, .., m, \sigma \in R_+\} \cup \{tlen/tlen \in R_+ \cup \{0\}\}$; ν is the time interval after that the first active station will finish his service, i.e. a new client may be served; $(ptime, j, k, \sigma, tl)$ means that the working station j, has been served one *k*-class client, *ptime*, respectively σ being the moment of the service finishing, respectively the length of the service and *tl* is the length of one laziness time. 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 func-

tion. *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 $(i, qtime, \sigma)$, 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;

$$tl = \begin{cases} qtime - ptime, & if\ D^{DP} = \{\}, pf \in \{1,2,..,n\}\ and\ s_j = (0,0,\infty); \\ 0, & otherwise \end{cases}$$

When the input is "*new-arriv*", this message is sent to the executive and his efect is: *ql:=ql+1* and if $ql/nbp \geq lmax\ and\ nbp < n$ the station $P_{r+1} = min\{j/j\ not\ in\ D^{DP}\}$ will became 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) output message, with $\nu = 0$, is generated.

Internal transition. Let $\Delta = min\{ta_i(s_i) - e_i/i \in D^{DP}\}$, $Im = \{r/r \in D^{DP}, ta_r(s_r) - e_r = \Delta\}$, $i^* = sel^{DP}(Im) = min\{r/r \in Im\}$, 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.:

- $ptime := ptime_{i^*}$;
- *tlen:=tl* and *tl:=0*;
- if $ql/nbp \geq lmin$, then $pf = i^*\ and\ \nu = \Delta$;
- if $ql/nbp < lmin\ and\ nbp > 1$ then $D^{DP} := D^{DP} - \{i^*\}\ and\ \nu = \infty$;
- if $ql/nbp < lmin\ and\ nbp = 1$ then $pf = i^*,\ D^{DP} := \{\}\ and\ \nu = 0$.

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 $s'_Q = \delta_{int_Q}(s_Q), e'_Q = 0, s'_P = \delta_{ext_P}(s_P, (i, qtime, \sigma), e), e'_P = 0$.

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 $s_Q^{'} = \delta_{ext}^{'}(s_Q, (\nu), e)), e_P^{'} = 0, s_P^{'} = \delta_{int_P}(s_P), e_P^{'} = 0$. The model outputs are (k, τ), respectively $(j, k, \sigma, ptime, tlen)$, being sent 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: $X_T = \{(i, \tau)/i = 1,..,m; \tau \in R_+ \cup \{0\}\} \cup \{(i, k, \sigma, ptime, tlen)/i = 1,..,n; k = 1,..,m; \sigma, ptime, tlen \in R_+ \cup \{0\}\}$. A state of this model is of form $s_T = (Tt, Ttl, Tst, Twt, Nss, Nsc)$, where: *Tt* is the value of the clock; *Ttl* is the total laziness time; *Tst* is a bi-dimensional vector, *Tst(i,k) (i=1,..,n; k=1,..,m)*, means the total working time of i station in order to serve the clients of k class; *Twt* is a vector *Twt(i) (i=1,..,m)* is the total waiting time in the queue of the i class clients; *Nsc* is a vector, *Nsc(i) (i=1,..,m)* contains the total numbers of the served customers of i class; *Nss* is a vector, *Nss(i) (i=1,..,n)* contains the total numbers of the served customers by the i working station. This model is passive, so only external transitions are performed. When it receives an input of form (i, τ) the model performs an external transition, as follows: $Twt(i) := Twt(i) + \tau$. If the input is of the form $(i, k, \sigma, ptime, tlen)$, then $Tt := ptime; Tst(i, k) := Tst(i, k) + \sigma; Nss(i) := Nss(i) + 1, Nsc(k) := Nsc(k) + 1, Ttl := Ttl + tlen$. Because the model is passive, $ta_T(s_T) = \infty$.

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 clients, $MTw(k) = \frac{Twt(k)}{Nsc(k)}, k = 1,..,m$;

- The average of the service time of the k class clients, $MTsc(k) = \frac{Tsc(k)}{Nsc(k)}$, where

$$Tsc(k) = \sum_{j=1}^{n} Tst(j, k), k = 1,..,m$$

- The average of the number of k class clients from the queue, $MLQ(k) = \frac{Twt(k)}{Tt}, k = 1,..,m$;

- The average of the length of the queue,

$$MLQ = \sum_{i=1}^{m} MLQ\,(i)$$

- The average of the stations laziness time, $MTtl = \frac{Ttl}{Tt}$
- The average of the service time of the i station, $MTts\,(i) = \frac{Tts(i)}{Nss(i)}$.
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. 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 simulators. 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.

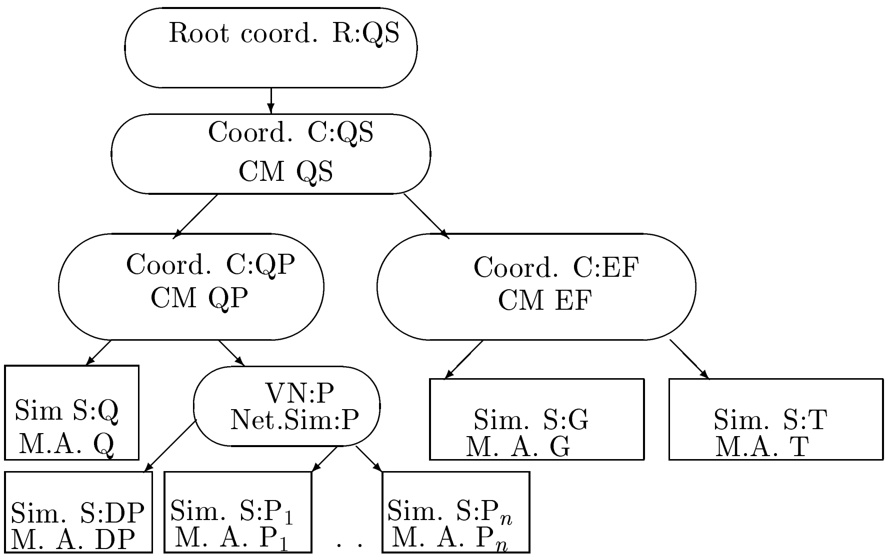

Figure 2: The abstract simulator structure and hierarchical construction (M.A.=Atomic Model, C.M=Coupled Model with fixed structure, V.N.M=Variable structure Network Model)

A very important aspect of the simulation of the DEVS models is the communication between simulators. The sending of the messages between them makes this. Indifferently of the type of the abstract simulator, there is five message types; a i-message corresponds to establish an initial state; a *-message, respectively a x-message, received from a coordinator at the immediately superior level, is showing to perform 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 receive (i,t)

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

send (done,t) to C:QP.

When receive (,t)*

done:=false;

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

else

$i := 1;$

While $\mathrm{i} \leq \; m \; and \; q_i \; \neq \; \Lambda \; i := i + 1;$

done:=false;

$if i \leq m$ *then*

k:=i; $tn_Q := tn_Q + \theta;$

$send((k, \tau_1^k), (k, tn_Q, \sigma_1^k)) \rightarrow \; C : QP;$

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

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

else "Error:";

send (done,t) to C:Q .

When receive $((i, gtime, \sigma), t)$;

done:=false;

$if\ not(tl_Q \le t \le tn_Q)\ \ then\ 'Error - Bad\ synchronisation'$

else

$s_Q := \delta_{ext_Q}(s_Q, t - tl_Q, (i, gtime, \sigma))$*; { $(0, \sigma)$ get on in the queue }*

done:=true; send (done,"new arriv",t) to C:QP.

When receive $((\nu), t)\ \theta := \nu$*; done:=true;*

send (done) to C:QP

end .

The abstract simulator of the variable structure DEVS model P is a network simulator. Firstly, the simulator associated of every active working station j, $j \in \{1, .., n\}$ is presented.

When receive (i,t) ;

$tl_j := t; tn_j := \infty; done := true;$

send (done,t) to NS:P.

When receive$((k, qtime, \sigma), t)$

done:=false;

$if\ not(tl_j \le t \le tn_j)\ then\ 'Error - Bad\ synchronisation'$

else

$s_j := \delta_{ext}(s_j, t - tl_j, (k, qtime, \sigma)); \{s_j = (k, qtime, \sigma)\}$

$tl_j := t; tn_j := tl_j + \sigma; done := true;$

send (done) to NS:P;

*When receive (*,t)*

done:=false;

$If\ t \ne tn_j\ then' Error - Bad\ synchronisation'$

Else

$tl_j := t; tn_j := \infty; ptime_j := ptime_j + \sigma;$

$s_j := \delta_{int_j}(s_j); done := true\ \{s_j = (0, \infty, 0)\}$

$send(done, (j, ptime_j, k_j, \sigma_j)) toNS : P;$

send (done) to NS:P.

The simulator S:DP of the executive

When receive (i,t) ;

$D^{DP} := \{1\}$*;send(i,t) to S:1;*

Wait Until(done) from S:1;

$tl_{DP} := t; tn_{DP} := \infty; ql := 0; nbp := 0; pf := 1; TTl := 0; read\,(L, l)\,;$

send (done) to NS:P.

When receive ("new arriv",t) ;

$tl_{DP} := t; tn_{DP} := \infty; ql := ql + 1;$

if $(ql/nbp > L)$ *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 $\nu := \infty$;

done:=true;send(done,ν) to NS:P.

When receive (i^*,t)

$tl_{DP} := t; tn_{DP} := \infty;$

if $(ql/nbp < l)$ *then*

if $(nbp > 1)$ *then* $D^{DP} := D^{DP} - \{i^*\}; nbp := nbp - 1; \nu := \infty$

else $pf := i^*; \nu := \infty;$

else $pf := i^*; \nu := 0;$

send(done,ν) to NS:P;.

The network simulator NS:P of model P

When receive (i,t)

send (i,t) to DP; WaitUntil done from DP;

$tl_P := t; tn_P := tn_{DP}$; *Done:=true; Send (done,t) to C:QP;*

When receive ("new arriv",t)

$if\ t \notin [tl_P, tn_P]\ then\ ERROR;$ *end.*

send ("new arriv",t) to S:DP;

$WaitUntil\ \ (done, \nu, t)\ \ from\ S : DP;$

$tl_P := t; tn_P := min\{tn_i / i \in D^{DP} \cup \{DP\}\};$

send(done,ν,t) to C:QP;

When receive ((i,qtime,σ),t)

$if\ t \notin [tl_P, tn_P]\ then\ ERROR;$ *end.*

send ((i,qtime,σ),t) to S:DP;

WaitUntil $(done, \nu, t)$ *from* $S : DP;$

$tl_P := t; tn_P := min\{tn_i / i \in D^{DP} \cup \{DP\}\};$

send(done,t) to C:QP;

*When receive (*,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(i^*,t) to S:DP; WaitUntil(done,ν) from S:DP:

send(done,$(i^*, k_{i^*}, \sigma_{i^*}, ptime_i{}^*), (\nu)$) 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 receive (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 receive (i,t)

read(Nevs) $\{$*Nevs= number of served clients* $\}$*;*

$tl_T := t; tn_T := \infty; Tst := 0n, m; Twt := 0m; Ttlen := 0$

Tss:=0n; Tsc:=0m; Nss:=0m;Nsc:=0n;evs:=0;

when receive $((k, \tau, t)$

$done := false;$

if $t \geq tl_T$*then* $done := true;\ Twt(k) := Twt(k) + \tau;$

send (done) to C:EF;

$when\ receive((i, k, \sigma, ptime, Tlen), t)$

done:=false;stopsim:=false;

$if\ t \geq tl_T\ then$

$done := true;\ Tt := ptime;\ tl_T := t;\ Nss(i) := Nss(i) + 1;$

$Nsc(k) := Nsc(k) + 1;\ Tst(i, k) := Tss(i, k) + \sigma;$

$evs := evs + 1;\ Twt(k) := Twt(k) + \tau; Ttlen := Ttlen + tlen\ ;$

if evs=Nevs then stopsim:=true;

send (done,stopsim) to C:EF

When receives(,t)*

Mqueue:=0;MTlen:=Ttlen/n;

for k:=1 to m do

$MTw(k) := Twt(k)/Nsc(k); MLq(k) := Twt(k)/Tt;$

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

$MTsc(k) = \frac{\sum_{j=1}^{m} Tst(j,k)}{Nsc(k)};$

for $i :=$ *1 to n do*

MTs(i)=$\sum_{j=1}^{n} Tss\,(i,j)\ /Nss(i);$

Send (done, (MTw(k), MTsc(k), MLq(k)); k=1,..,m;

Mtlen, MTs(i);i=1,..,n), 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

The advantages of Object Oriented Simulation are presented in [2, 3]. In [7] I have fulfilled a simulation product, based on the DEVS approach and OO ideas, in order to study the priority queuing system. In the following, the results obtained by simulation, for two queuing models of head of line type are presented. In order to achieve the model validation, these results are compared with those analytically obtained and I have remarked that they are approximately equals.

5.1 The model with one station and two priority class.

Because we have only one working station, this model is identical with the fixed number of station of head of line type with one station and two priority class. Considering the *arrivals simulated number=10000*, I have been obtained:

- The average of the waiting time in the queue of 1-class clients= 0.635;

- The average of the waiting time in the queue of 2-class clients= 1.319;
- The average of the length of the queue =0.491;
- The average of the serving time of the 1-class client=0.990;
- The average of the serving time of the 2-class client=0.996;
- The average of the serving time of the working station =0.993;
- The using coefficient of the working station = 0.498.

Analytically, in [9, 11] this model is studied. Using the formulae presented in these books, in [8] I have obtained aproximatly equal results with those above presented.

5.2 The system with two priority class and two service stations

Considering, *Lmax*, respectively *Lmin* thersholds as being *Lmax=4, Lmin=2* and the *number of simulated arrivals=10000*, I have obtained the following results:
- The average of the waiting time in the queue of 1-class client = 0.693;
- The average of the waiting time in the queue of 2-class client = 2.616;
- The average of the length of the queue = 1.673;
- The average of the serving time of the 1 class clients =0.983;
- The average of the serving time of the 2 class clients =0.972;
- The average of the 1-station serving time = 0.991;
- The average of the 2-station serving time = 0.959;
- The using 1-station coefficient =0.568;
- The using 2-station coefficient =0.397;
- The average of the laziness coeficient of 1-station=0.432;
- The average of the laziness coeficient of 2-station=0.603;
- The average of the serving time of all clients = 0.978;
- The average of the laziness coeficient of all stations=0.518.

Remark In [8] I have simulated the corresponding model with fixed number of working stations. I am remarking 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.

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 Waiting Systems of Head of Line Type: An Object Oriented Formal 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