Computer Science Journal of Moldova, vol.**1**, no.1(1), 1993

The Object Oriented Programming for Queuing System

I.Grama G.Mishkoy

Abstract

The object oriented programming approach for priority service systems with orientation is developed. It provide fast evaluation of the characteristics.

1 Introduction

Priority systems with orientation have been the subject of most intensive investigation for the past decade. In particular in [1] a class of systems of this type with non-zero orientation time of the service device as well as with its various strategies in free state have been studied. The necessity and significance of the analysis of these systems have been pointed out by various scientists (see, for example [2], [3], [4], [5]) and have been caused by the entire set of requirements to the development of priority systems theory (the need of studying more common models and obtaining of new more common mathematical results) and by actual practice need.

In the applied aspect a number of problems of real-time systems, information and computation systems and a number of problems of other systems are well described by models of these systems.

The main peculiarity of the mentioned models is that they allow taking into account time losses during different kinds of works, switchings and so on, taking place in real systems and having an accidental character. For example, these are losses within the switching of

computer process in scheduling algorithms, within the information exchange while requesting the memory of exchange, within writing and restoring in the interrupt computer systems, etc.

New approaches and conception for solving such topical applied problems can be suggested on the basis of the results of this class of systems. For example, on the basis of the results of priority systems with orientation the conception of functional designing of computation complexes has been worked out and realized [5]. New technical solution can be suggested too.

It must be noted that the mathematical results were obtained in terms of the generating function, the Laplace and the Laplace-Stieltjes transforms and some recurrent functional equations. For practical needs it is necessary to create the software which allows us to evaluate the characteristics of the priority systems. It includes methods solving the recurrent functional equations, inverting the Laplace and Laplace-Stieltjes transform and so on.

Earlier a version of this software was worked out in [8]. The aim of the paper is to present a new approach to the priority systems software based on the object oriented programming (OOP).

2 Analytical means

According to the mathematical theory of priority systems it is supposed that the switching process from one class to another one takes place instantly (for example, see [6]). But this assumption is not fulfilled for a considerable part of real systems since switching always takes some time. The systems which take into account the switching process are called the priority service systems with orientation (PSSO) and the switching time is called orientation time.

We shall present some results describing the behavior of the PSSO. It should be pointed out that difficulties arise with their implementation for practical needs, since there are not efficient algorithms for evaluating its characteristics.

While making the statement, we shall keep the terminology and notations adopted in [1] and formulate the results for $M_r|G_r|1$ priority

model involving orientation and absolute priority.

The arrival of the higher priority requests in the device, busy with queuing orientation or servicing the lower priority request, interrupts both the orientation and the service. It is supposed that when the PSSO becomes free of the higher priority requests the interrupted orientation will be continued, while the interrupted service is restarted.

As far as the strategy of the device in the idle state is concerned, let us assume that its orientation gets instantly annulled (reset) as soon as the busy period is completed.

2.1 Distribution of the busy period

Let us denote by $B_i(t)$, $C_j(t)$ and $\Pi(t)$ the distribution functions of duration of service of requests of the i-th priority class, duration of orientation of the device for servicing the requests of j-th class, $i \neq j$, $i = 1, ..., r$ and busy period, respectively. Let a_i be the parameter of the arriving Poisson flow of priority i and $\sigma_k = a_1 + ... + a_k$, $\sigma_0 = 0$, $\sigma = \sigma_r$. Let $\beta_i(s)$, $c_j(s)$, $\pi(s)$ be the Laplace-Stieltjes transforms of the distribution functions $B_i(t)$, $C_j(t)$, and $\Pi(x)$ respectively i.e.

$$\beta_i(s) = \int_0^\infty e^{-st} dB_i(t), ... \quad .$$

Statement 1 *The Laplace-Stieltjes transform of the distribution function of a busy period is determined from the system of recurrent functional equations.*

$$\begin{aligned}
\sigma_k \pi_k(s) &= \sigma_{k-1}\pi_{k-1}(s + a_k) \\
&\quad +\sigma_{k-1}\{\pi_{k-1}(s + a_k(1 - \bar{\pi}_{kk}(s))) - \pi_{k-1}(s + a_k)\} \quad (1) \\
&\quad \times\nu_k(s + a_k(1 - \pi_k(s))) + a_k\pi_{kk}(s), \\
\pi_{kk}(s) &= \nu_k(s + a_k(1 - \bar{\pi}_{kk}(s)))\bar{\pi}_{kk}(s), \quad (2) \\
\bar{\pi}_{kk}(s) &= h_{k-1}(s + a_k(1 - \bar{\pi}_{kk}(s))), \quad (3) \\
h_k(s) &= \frac{\beta_k(s + \sigma_{k-1})}{1 - \frac{\sigma_{k-1}}{s+\sigma_{k-1}}[1 - \beta_k(s + \sigma_{k-1})]\nu_k(s)}, \quad (4) \\
\nu_k(s) &= c_k(s + \sigma_{k-1}[1 - \pi_{k-1}(s)]). \quad (5)
\end{aligned}$$

Functions $\pi_k(s)$,...,$\nu_k(s)$ included in expression (1)-(5) are the Laplace-Stieltjes transforms of distribution functions $\Pi_k(t)$,...,$N_k(s)$ of the supplementary intervals of the k-th period,..., the k-th orientation cycle having rather determined independent meaning. Thus, (4) is nothing but the distribution of the total time of the priority request k occurrence in the device. Let us denote by β_{k1}, c_{i1}, π_{k1},..., ν_{k1} the first moments of the distribution functions $B_k(t)$, $C_k(t)$, $\Pi_k(t)$,..., $N_k(t)$, respectively. Let

$$\rho_k = \sum_{i=1}^{k} a_i b_i,$$

where

$$\begin{aligned} b_1 &= \frac{\beta_{11} + c_{11}}{1 + a_1 c_{11}}, \\ b_i &= \Phi_1 \dots \Phi_{i-1} \frac{1}{\sigma_{i-1}} [\frac{1}{\beta_i(\sigma_{i-1})} - 1](1 + \sigma_{i-1} c_{i1}), \\ \Phi_1 &= 1, \quad \Phi_i = 1 + (\sigma_i - \sigma_{i-1}\pi_{i-1}(a_i))c_{i1}, \quad i = 2, ..., k. \end{aligned}$$

Statement 2 *If* $\rho_k < 1$ *then*

$$\begin{aligned} \sigma_k \pi_{k1} &= \frac{\Phi_2 \dots \Phi_k + \rho_{k-1}}{1 - \rho_k}, \\ \bar{\pi}_{k1} &= \frac{b_k}{1 - \rho_k}, \\ h_{k1} &= \frac{b_k}{1 - \rho_{k-1}}, \\ \nu_{k1} &= \frac{\Phi_2 \dots \Phi_{k-1}}{1 - \rho_k} c_{k1}. \end{aligned}$$

2.2 Probabilities of the $\rightarrow j$-state

In the process of functioning of a queuing priority system with orientation the servicing device can be in one of the following states:

- busy with servicing a request,
- busy with orientation for servicing,

- free from orientation for servicing or servicing itself.

In the case when the device is busy with servicing, the question arises as what priority class of requests is servicing and in the case if the device is busy with orientation to what priority class of requests is it orienting in the time under consideration ?

Let $\vec{P}_j(s)$ denote the probability that at the time instant $t \in [0, \infty)$ the device is busy with orientation for servicing of a request of the j-th priority class, $j = 1, ..., r$. Furthermore, let us denote

$$\vec{p}_j(s) = \int_0^\infty e^{-st}\, \vec{P}_j(t)dt.$$

Statement 3 *The Laplace transform $\vec{p}_j(s)$ satisfies the following equality*

$$\vec{p}_j(s) = \frac{\sigma\ \vec{\pi}_j(s)}{s + \sigma - \sigma\pi(s)},$$

where

$$\begin{aligned}
\sigma_k\ \vec{\pi}_k(s) &= \{\psi_j(s)\gamma_{j-1}(s) \\
&\quad + \frac{G_j(s)\sigma_{j-1}\pi_{j-1}(s)\psi_j(s)Q_j(s)}{1 - h_j(s)}\} \\
&\quad \times \prod_{i=j+1}^{k} \{1 + \psi_i(s)\gamma_{i-1}(s) \\
&\quad + \frac{[1 + \sigma_{i-1}\pi_{i-1}(s)\psi_i(s)]G_i(s)Q_i(s)}{1 - h_i(s)}\}, \qquad j < k, \\
\sigma_k\ \vec{\pi}_k(s) &= \{\psi_k(s)\gamma_{k-1}(s) \\
&\quad + \frac{G_k(s)\sigma_{k-1}\pi_{k-1}(s)\psi_k(s)Q_k(s)}{1 - h_k(s)}\}, \qquad j = k, \\
Q_j(s) &= \gamma_{j-1}(s)\nu_j(s) - \sigma_{j-1}\pi_{j-1}(s + a_j) - \sigma_j\pi_j(s), \\
\gamma_{i-1}(s) &= \sigma_{i-1}[\pi_{i-1}(s) - \pi_{i-1}(s + a_i)] + a_i, \\
\psi_j(s) &= \frac{1 - c_j(s + \sigma_{j-1}[1 - \pi_{j-1}(s)])}{s + \sigma_{j-1}[1 - \pi_{j-1}(s)]} \\
G_j(s) &= \frac{1 - \beta_j(s + \sigma_{j-1})}{s + \sigma_{j-1}[1 - \beta_j(s + \sigma_{j-1})]\pi_{j-1}(s)\nu_j(s)}.
\end{aligned}$$

Functions $\pi_j(s)$, $\nu_j(s)$, $h_j(s)$ are determined from the expressions (1)-(5).

2.3 Probabilities of $*j$-state

Let ${}^*P_j(t)$ be the probability that at the instant t the device is busy with servicing the requests of the class j,$j = 1, ..., r$. Let

$$ {}^*p_j(s) = \int_0^\infty e^{-st}\, {}^*P_j(t)dt. $$

Statement 4 *The Laplace transform ${}^*p_j(s)$ satisfies the following equality*

$$\begin{aligned}
{}^*p_j(s) &= \frac{\sigma\ {}^*_j\pi(s)}{s+\sigma-\sigma\pi(s)}, \\
\sigma_k\ {}^*_j\pi_k(s) &= \frac{G_j(s)\psi_j(s)Q_j(s)}{1-h_j(s)} \\
&\quad \times \prod_{i=j+1}^{k} \{1+\psi_i(s)\gamma_{i-1}(s) \\
&\quad + \frac{[1+\sigma_{i-1}\pi_{i-1}(s)\psi_i(s)]G_i(s)Q_i(s)}{1-h_i(s)}\}, \quad j<k, \\
\sigma_k\ {}^*_j\pi_k(s) &= \frac{G_k(s)\psi_k(s)Q_k(s)}{1-h_k(s)}, \quad j=k.
\end{aligned}$$

Functions $Q_j(s)$, ..., $G_j(s)$ are determined above.

Let ${}^*\mathbf{P}_j$ and ${}^{\rightarrow}\mathbf{P}_j$ be the stationary probabilities of the $\rightarrow j$-th and $*j$-th state.

Statement 5 *If $\rho_r < 1$ then*

$$ {}^*\mathbf{P}_j = \frac{\sigma\ {}^*_j\pi(0)}{1+\sigma\pi_1}, $$

$$ {}^{\rightarrow}\mathbf{P}_j = \frac{\sigma\ {}^{\rightarrow}_j\pi(0)}{1+\sigma\pi_1}, $$

where $\pi_1 = \pi_{r1}$ is determined (at $k = r$) from statement 2.

3 Object oriented programming for queuing systems. The basic objects.

The basic principles of OOP for queuing system will be presented in the sequel. Through this compartment we follow the convenience adopted in the programming language PASCAL 6.0 stating that any object has at most one ancestor.

We begin the description of the queuing system in terms of object oriented programming by finding its most simple structure unit. From the Probability Theory's point of view the simplest structure element in some queuing system is the random variable (r.v.). This is so because all quantities involved in the description of the model of the queuing system are some characteristics of positive r.v..

For the sake of definiteness we worked out two examples. So the equations (1)-(5) are written in terms of the Laplace-Stieltjes transforms of the distribution functions (d.f.) of such time periods as π_k, π_{kk}, $\bar{\pi}_{kk}$, h_k, ν_k periods. In their turn all these time periods are positive r.v.. For the second example we have the probabilities of $\rightarrow j$-state $\vec{P}_j(s)$ and $*j$-state $\overset{*}{P}_j(s)$ which are understood as the distribution of the r.v. with the discrete values $1, 2, ..., r$.

3.1 Object type RV

In this section we describe the object type RV corresponding to the abstract r.v. with the continuous d.f..

We begin by selecting three different possibilities to define a r.v.. The first one is to define r.v. by means of their d.f. $DistrF(t)$. The second one is to describe a r.v. by using the density function (ds.f.) $DensityF(t)$ if it exists. Interdependence of these two methods of defining a r.v. are given by the relations

$$DistrF(t) = \int_0^t DensityF(s)ds,$$

$$DensityF(s) = DistrF^{'}(t).$$

For the third possibility we recall that the basic equations (1)-(5) describing PSSO are in terms of the Laplace-Stieltjes transform of d.f. of some r.v.. Therefore we have to describe a r.v. also in terms of the Laplace-Stieltjes transform $LaplaceF(t)$ of their d.f.. In order to establish the connection between two functions $DistrF(t)$ and $LaplaceF(t)$ we have to mention that there exists a one-to-one application from the set of d.f. onto the set of their Laplace-Stieltjes transforms. This gives us the possibility to describe completely a r.v. knowing only the Laplace-Stieltjes transform as well.

The above mentioned allows us to understand a r.v. as the collection of three functions: $DistrF(t)$, $DensityF(t)$, $LaplaceF(t)$. For the reasons which we will explain just now it is convenient to specify the function $DistrF(t)$ to be the inversion formula of the Laplace-Stieltjes transform $LaplaceF(t)$

$$DistrF(t) = \frac{1}{2\pi i}\int_C \frac{LaplaceF(s)}{s}e^{st}ds,$$

C – being the integration contour $C = \{z$ is complex number with $Re\ z > 0\}$, the $DensityF(t)$ to be the inversion formula of the Laplace transform $LaplaceF(t)$

$$DensityF(t) = \frac{1}{2\pi i}\int_C LaplaceF(s)e^{st}ds,$$

C – being the above integration contour and $LaplaceF(t)$ to be the Laplace-Stieltjes transform formula

$$LaplaceF(t) = \int_0^\infty e^{-st}dDistrF(s).$$

We argue such choice in the following manner. In case we have known for instance only the r.v.'s function $LaplaceF(t)$ then we cover the inherited method $LaplaceF(t)$ by new a method which can explicitly calculate it while for unknown characteristics $DistrF(t)$ and $DensityF(t)$ the inherited methods will provide approximation formulae.

So we introduce the object type `RV` corresponding to the r.v. as follows:

Object type

$$\texttt{RV} = \begin{cases} \textit{Ancestor}: & \\ \textit{Fields}: & \\ \textit{Methods}: & \texttt{Init;} \\ & \texttt{Done;virtual;} \\ & \texttt{DistrF(t);virtual;} \\ & \texttt{DensityF(t);virtual;} \\ & \texttt{LaplaceF(t);virtual;} \end{cases}$$

Description:

- `Init, Done`: empty constructor and detractor,
- `DistrF(t)`: method calculating the inversion of the Laplace-Stieltjes transform of `LaplaceF(t)`,
- `DensityF(t)`: method calculating the inversion of the Laplace transform of `LaplaceF(t)`,
- `LaplaceF(t)`: method calculating the Laplace-Stieltjes transform of `DistrF(t)`.

The just described object type `RV` does not represent a really existing r.v.. We only need the object type `RV` to construct the object types related with the r.v. of the desired kind to be used in modeling the queuing system.

We shall present an example of an object type corresponding to the r.v. with the known Laplace-Stieltjes transform $f(t, a)$, a being some parameter:

Object type

$$\texttt{TMyRV} = \begin{cases} \textit{Ancestor}: & \texttt{RV;} \\ \textit{Fields}: & \texttt{a;} \\ \textit{Methods}: & \texttt{LaplaceF(t);virtual;} \end{cases}$$

Description:

- `a` is the value of parameter of $f(t, a)$,

- `LaplaceF(t)` $= f(t, a)$.

If variable `MyRV` is of type `TMyRV` then to access the value of its d.f. `DistrF(t)` we write simply `MyRV.DistrF(t)` to call the implemented method inverting the Laplace-Stieltjes transform of `MyRV.LaplaceF(t)`.

Now we proceed describing some types of useful r.v.. We present the description of the object types related to the r.v. with exponential and Erlang d.f.. The structures of these object types are as follows:

Object type

$$\texttt{ExpRV} = \begin{cases} \textit{Ancestor}: & \texttt{RV;} \\ \textit{Fields}: & \texttt{a;} \\ \textit{Methods}: & \texttt{DistrF(t);virtual;} \\ & \texttt{DensityF(t);virtual;} \\ & \texttt{LaplaceF(t);virtual;} \end{cases}$$

Description:

- `a`: parameter,
- `DistrF(t)`$= 1 - e^{-at}$,
- `DensityF(t)`$= ae^{-at}$,
- `LaplaceF(t)`$= a/(a + t)$.

Object type

$$\texttt{ErlRV} = \begin{cases} \textit{Ancestor}: & \texttt{RV;} \\ \textit{Fields}: & \texttt{a,k;} \\ \textit{Methods}: & \texttt{DistrF(t);virtual;} \\ & \texttt{DensityF(t);virtual;} \\ & \texttt{LaplaceF(t);virtual;} \end{cases}$$

Description:

- `a, k`: parameters,
- `DensityF(t)` $= a^k \frac{t^{k-1}}{(k-1)!} e^{-at}$,
- `LaplaceF(t)` $= \frac{a^k}{(a+t)^k}$.

3.2 Object type Queue

In this section we describe the object type related with the exponential input of requests. To this end let us explain the structure of input queue. The input queue is formed by all arriving requests while the PSSO is busy. It includes r flows corresponding to the priorities 1,...,r. The time length between two consecutive arrivals of requests of the same flow is the exponential r.v. with parameters a_i.

In order to define the flow we use the object type Flow with the following structure:

Object type

$$\texttt{Flow} = \begin{cases} Ancestor: & \texttt{ExpRV;} \\ Fields: & \\ Methods: & \texttt{SetIntensity(a);} \\ & \texttt{GetIntensity;} \end{cases}$$

Description:

- `SetIntensity(a)`: sets the value of the parameter of exponential r.v.,
- `GetIntensity`: gets the value of the parameter of exponential r.v..

Since the input queue contains r flows we can understand it as a collection of r r.v.. Corresponding to the input queue we introduce the object type `Queue`. Its structure is presented as follows:

Object type

$$\texttt{Queue} = \begin{cases} Ancestor: & \\ Fields: & \texttt{Flow;} \\ Methods: & \texttt{Init(r);} \\ & \texttt{Done(r);virtual;} \\ & \texttt{DistrF(k,t);virtual;} \\ & \texttt{DensityF(k,t);virtual;} \\ & \texttt{LaplaceF(k,t);virtual;} \\ & \texttt{SetIntensity(k,a);} \\ & \texttt{Intensity(k);} \end{cases}$$

Description:

- `Flow`: the pointer to the array of objects of type `Flow`,
- `Init(r)`: creates the dynamic array of dimension `r` of objects of type `Flow` and set the pointer to this array in `Flow`,
- `Done(r)`: disposes the dynamic array of dimension `r` of objects of type `Flow` at the pointer `Flow`,
- `DistrF(k,t)`, `DensityF(k,t)`, `LaplaceF(k,t)`: return the values `DistrF(t)`, `DensityF(t)`, `LaplaceF(t)` of the k-th flow,
- `SetIntensity(k,a)`, `Intensity(k)`: set and return the intensity a_k of the k-th flow.

3.3 Object types Orient and Service

This section deal with the object types related to the orientation and service periods.

Recall that before proceeding to serve the request of priority k the PSSO needs an orientation time period for preparing, if the previous request's priority differ of k. This time period is actually a positive r.v.. The orientation times corresponding to different priorities are distinct, so in order to describe the orientations process we need a collection of `r` r.v.. The object type `Orient` is created to keep this r.v. and to provide their treatment.

Object type

$$\texttt{Orient} = \begin{cases} \mathit{Ancestor}: & \\ \mathit{Fields}: & \texttt{Time;} \\ \mathit{Methods}: & \texttt{Init(r);} \\ & \texttt{Done(r);} \\ & \texttt{SetRV(k,OrientTime);} \\ & \texttt{DistrF(k,t);} \\ & \texttt{DensityF(k,t);} \\ & \texttt{LaplaceF(k,t);} \end{cases}$$

Description:

- `Time`: the pointer to the array of objects of type `RV`,
- `Init(r)`: creates the dynamic array of dimension `r` of pointers of type `RV` and sets the pointer to this array in `Time`,
- `Done(r)`: disposes the dynamic array of dimension `r` of objects of type `RV` at the pointer `Time`,
- `SetRV(k,OrientTime)`: sets the pointer `Time[k]` to point to the object of type `RV` wanted to be the k-th orientation time,
- `DistrF(k,t)`, `DensityF(k,t)`, `LaplaceF(k,t)`: return the values `LaplaceF(t)`, `DistrF(t)`, `DensityF(t)` of the k-th orientation time.

After the orientation the PSSO device proceeds to serve the request. Again the PSSO device need a random time period to serve the request. So a collection of `r` r.v. fully describe the service times. Object type `Service` is destined to handle these service times and is defined in the same way as object type `Orient`.

3.4 Object type Status

At this point we have to create an object for the state of the PSSO.

The PSSO state is determined by two parameters we call `Scheme` and `Regime` respectively (see [8]). The first one is a bidimensional vector with components `OrientDisc` and `ServiceDisc` each oh them describing the disciplines of orientation and service respectively. Parameter `Scheme` takes the values in the following table

$$\begin{array}{cccc} 1.1 & (1,2) & (1,3) & (1,4) \\ 2.1 & (2,2) & (2,3) & (2,4) \\ 3.1 & (3,2) & (3,3) & (3,4) \end{array}$$

Schemes (1,1),...,(2,3) represent six types of the absolute priority `Abs1`, `Abs2`, `Abs3`, `Abs4`, `Abs5`, `Abs6`. Schemes (3,1),...,(3,3) represent three

types of the semiabsolute priority `SemiAbs1`, `SemiAbs2`, `SemiAbs3`. Schemes (1,4),(2,4) represent two types of the semirelative priority `SemiRel1`, `SemiRel2`. Scheme (3,4) represents the relative priority `Rel`.

The parameter Regime set the behavior of the PSSO while the system is not busy. There are three alternatives `Reset`, `LookAhead`, `WaitMostProb`.

The object type `Status` provides a handle for the parameters `Dimension`, `OrientDisc`, `ServiceDisc` and `Regime`.

Object type

```
Status = { Ancestor :
           Fields :     Dimension;
                        OrientDisc; ServiceDisc;
                        Regime;
           Methods :    Init; Done;virtual;
                        SetDimension(Dim);virtual;
                        GetDimension;virtual;
                        SetServiceDisc(ServiceD);virtual;
                        GetServiceDisc;virtual;
                        SetOrientDisc(OrientD);virtual;
                        GetOrientDisc;virtual;
                        SetRegime(Reg);virtual;
                        GetRegime;virtual;
                        SetPriority(Prior);virtual;
                        GetPriority;virtual;
```

Description:

- `Dimension`: the numbers of priorities r,
- `OrientDisc`: keeps the orientation discipline,
- `ServiceDisc`: keeps the service discipline,
- `Regime`: keeps the parameter Regime,
- `Init`: installs the default values for all fields,
- `Done`: destroys the object,

- All other methods install or return the values of corresponding parameters.

3.5 Object type MG

Having defined the object types `Queue`, `Service`, `Orient` and `Status` we can construct now the object type `MG` containing the full background information on the PSSO.

Object type

$$\texttt{MG} = \begin{cases} \textit{Ancestor}: & \texttt{Status;} \\ \textit{Fields}: & \texttt{Queue;} \\ & \texttt{Service;} \\ & \texttt{Orient;} \\ \textit{Methods}: & \texttt{Init(Dim);} \\ & \texttt{Done;virtual;} \end{cases}$$

Description:

- `Queue`: the object of type `Queue`,
- `Service`: the object of type `Service`,
- `Orient`: the object of type `Orient`,
- `Init, Done`: call the `Init, Done` methods of the objects `Queue`, `Service`, `Orient` and `Status`.

3.6 Object type PSSO

Object type `PSSO` is introduced to provide a background for PSSO characteristics. It includes the procedures `Fast` and `FastIter` which are of crucial importance in the calculation of all characteristics.

Object type

$$\texttt{PSSO} = \begin{cases} \textit{Ancestor}: & \texttt{MG;} \\ \textit{Fields}: & \texttt{a, sigma;} \\ & \texttt{cnt, InitpkkDimension, Initpkk;} \\ & \texttt{pks, bpkk, pk1s1, pk1s2;} \\ \textit{Methods}: & \texttt{Fast(k,s);} \\ & \texttt{FastIter(k,s);} \\ & \texttt{n(k,s,x);} \\ & \texttt{h(k,s,x,y);} \end{cases}$$

Description:

- `a,sigma`: the arrays of input intensities and their partial sums,
- `cnt, InitpkkDimension, Initpkk`: some quantities for use in methods `Fast` and `FastIter`,
- `pks`, `bpkk`, `pk1s1`, `pk1s2`: the values $\pi_k(s)$, $\bar{\pi}_{kk}(s)$, $\pi_{k-1}(s+a_k)$, $\pi_{k-1}(s+a_k(1-\bar{\pi}_{kk}(s)))$ respectively,
- `Fast, FastIter, n, h`: produce background calculation for all PSSO characteristics,

3.7 Object types related with PSSO characteristics

We treat any PSSO characteristic as a r.v. related to some object of type `PSSO`. This allows the characteristic to access the fields and methods of the object for its own use. First we introduce the abstract type `PSSOChar` which include the common feature of all characteristics, i.e. the property to access the object of type `PSSO`.

Object type

$$\texttt{PSSOChar} = \begin{cases} \textit{Ancestor}: & \texttt{RV;} \\ \textit{Fields}: & \texttt{k;} \\ & \texttt{PSSOptr;} \\ \textit{Methods}: & \texttt{Init(PSSO);} \\ & \texttt{SetLevel(i);} \end{cases}$$

Description:

- `k`: selected priority,
- `PSSOptr`: pointer to the object of type `PSSO`,
- `Init(PSSO)`: sets `PSSOptr`=`PSSO`,
- `SetLevel(i)`: sets `k`=`i`,

Now we proceed defining object types corresponding to some concrete PSSO characteristics. To do this we only need to redefine the inherited method `LaplaceF` of the object type `PSSOChar` in the proper way.

First we define the object type of the characteristic $\pi_k(s)$.

Object type

$$\texttt{pk} = \begin{cases} Ancestor: & \texttt{PSSOChar;} \\ Fields: & \\ Methods: & \texttt{LaplaceF(t);} \end{cases}$$

Description:

- `LaplaceF(t)` runs the procedure `Fast(k,t)` of the object to which `PSSOptr` points and returns the value `pks`.

Now let us present the object type of the characteristic π_{kk}.

Object type

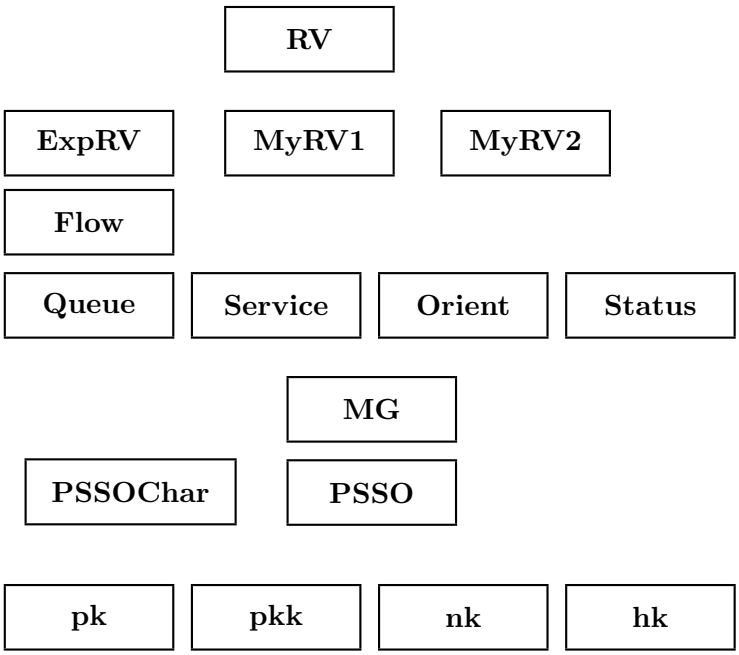
$$\texttt{pkk} = \begin{cases} Ancestor: & \texttt{PSSOChar;} \\ Fields: & \\ Methods: & \texttt{LaplaceF(t);} \end{cases}$$

Description:

- `LaplaceF(t)` runs the procedure `Fast(k-1,t)` of the object to which `PSSOptr` points and returns the value `n(k,t,pks)`.

For other characteristics the similar object types are available.

The connection between object types is presented in the figure below.



The thin line shows the origin of the object while the thick line shows what objects it contains .

4 Numerical results

The above object types were used to evaluate the characteristics π_k, π_{kk}, $\bar{\pi}_{kk}$, n_k, h_k. In the following tables we present some numerical results. The first number inside each column represents the value of the corresponding characteristic, the second one being the processing time in msec . All calculations were performed at the computer IBM PC 286.

In the first table we evaluate the Laplace-Stieltjes transform of the

d.f. corresponding to the characteristics.

t	π_k		π_{kk}		$\bar{\pi}_{kk}$		n_k		h_k	
.00001	1.000	61	1.000	61	1.000	61	1.000	22	1.000	16
.1	.8551	66	.8601	66	.9136	71	.9571	22	.9365	22
.5	.6039	44	.6111	44	.7441	44	.8467	17	.7788	22
1.23	.4156	33	.4205	27	.5956	33	.7300	11	.6253	11
3.45	.2088	22	.2102	22	.3986	22	.5442	11	.4156	11
5.67	.1310	16	.1316	17	.3066	16	.4415	06	.3176	06
10.12	.0670	16	.0671	16	.2124	17	.3233	05	.2182	06
20.34	.0250	11	.0250	16	.1259	17	.2015	06	.1281	05

In the second table the d.f. corresponding to the characteristics are evaluated .

t	π_k		π_{kk}		$\bar{\pi}_{kk}$		n_k		h_k	
.00001	.000	038	.000	039	.000	039	.000	022	.000	022
.1	.058	104	.058	110	.251	110	.396	049	.256	049
.5	.405	159	.408	165	.627	159	.798	072	.660	077
1.23	.646	214	.656	214	.799	220	.903	099	.844	099
3.45	.845	303	.855	307	.919	302	.973	132	.958	132
5.67	.911	363	.918	357	.955	357	.990	149	.984	149
10.12	.963	429	.965	428	.982	429	.998	165	.997	165
20.34	.992	510	.992	511	.996	511	1.00	182	1.00	187

References

[1] Klimov, G.P. and Mishkoy, G.K. (1979): Priority service systems with orientations. Moscow State University, (Russian).

[2] Jaiswal, N.K. (1968): Priority queues. Academic Press, New York.

[3] Bronstein, O.I. and Dukhovny, I.M. (1976): Priority models in computer systems. Nauka, Moscow, (Russian).

[4] Lipaev, V.V. (1979): Allocation of resources in computer systems. Statistica, Moscow, 1979, (Russian).

[5] Kabalevsky, A.N. (1986): Personal computers. Nauka, Moscow, (Russian).

[6] Gnedenko, B.V. et al. (1973): Priority service systems. Moscow State University, (Russian).

[7] Mishkoy, G.K. (1990): Priority queuing involving orientation and the problems of their software implementation. Computers Math.Applic. vol 19, no1, pp. 109-113.

[8] Mishkoy, G.K. etc. (1990): Software for priority systems with orientation. Stiintsa, Kishinev, (Russian).

I.Grama, G.Mishkoy
Institute of Mathematics,
Academy of Sciences of Moldova,
5 Academiei str.,
Chişinău, 277028, Moldova

Received September 15, 1992