СТАТИСТИЧЕСКИЕ ПРОБЛЕМЫ УПРАВЛЕНИЯ

ВЫПУСК

33

ВИЛЬНЮС, 1979

ИНСТИТУТ МАТЕМАТИКИ И КИБЕРНЕТИКИ АН ЛИТОВСКОЙ ССР
ИНСТИТУТ ФИЗИКО-ТЕХНИЧЕСКИХ ПРОБЛЕМ ЭНЕРГЕТИКИ АН ЛИТ. ССР
ЛИТОВСКАЯ ТЕРРИТОРИАЛЬНАЯ ГРУППА НКАУ СССР
МЕЖДУНАРОДНОЙ ФЕДЕРАЦИИ ПО АВТОМАТИЧЕСКОМУ УПРАВЛЕНИЮ

СТАТИСТИЧЕСКИЕ ПРОБЛЕМЫ УПРАВЛЕНИЯ

Выпуск 33

ТЕОРИЯ ДИНАМИЧЕСКИХ СИСТЕМ

(Материалы к семинару при Институте математики
и кибернетики АН Литовской ССР "СТАТИСТИЧЕСКИЕ
ПРОБЛЕМЫ УПРАВЛЕНИЯ")

Под редакцией В. Слявинскас

Институт математики и кибернетики Академии наук Литовской ССР
Вильнюс
1979

При Институте математики и кибернетики АН Литовской ССР постоянно действует республиканский научный семинар, состоящий из следующих шести секций:

1. Дифференциальные уравнения и их применение
2. Математические методы в социальных науках
3. Автоматизация процессов планирования и управления
4. Применение теории вероятностей и математической статистики
5. Статистические проблемы управления
6. Теория оптимальных решений.

Материалы каждой секции издаются в виде отдельной серии выпусков, нумеруемых продолжающейся порядковой нумерацией в пределе этой секции. Издаются не периодически, а по мере отбора материала по данной целевой тематике.

Для приобретения изданий обращаться по адресу:
232600, г. Вильнюс,
ул. К. Пожелос, 54,
Институт математики и кибернетики АН Лит.ССР
Сектор информации и патентов

С 3 05 02 - I I26 - Ž - 79
М86I-78

В настоящем сборнике рассматриваются принципы построения динамической модели с конечным и бесконечным множеством состояний.

Rinkinyje nagrinėjami dinaminių modelių su baigtine ir begaline būsenų aibe sudarymo principai.

The problem of the dynamical models' construction with finite and infinite sets of states is considered in this issue.

С О Д Е Р Ж А Н И Е

Бурачас А.,Габриёлавичюс А. Экономический анализ и теория систем 9

Сливинскас В. Методы минимальной реализации линейных динамических стационарных систем. Введение 3I

Иазеолла Д., Мартинелли Э., Праневичюс Г. Метод аналитического представления имитационных моделей дискретных событий 75

Тамулинас Б. Синтез правил поиска расписаний в системах обслуживания 9I

Рефераты II3

СТАТИСТИЧЕСКИЕ ПРОБЛЕМЫ УПРАВЛЕНИЯ
1979 ВЫПУСК 33

УДК 681.3:001.57

A METHOD FOR THE ANALYTICAL REPRESENTATION OF DISCRETE EVENT SIMULATION MODELS

Giuzepe IAZEOLLA*, Enrico MARTINELLI**, Henrikas PRANEVITCHIUS***

A mathematical style of description of simulation models is introduced as an alternative to more traditional methods, like flowgraphs and programming lanquaqes, which are less suited to represent the system at intermediate levels of development of the model. Understandability of the model, machine generation of simulation code and proof of correctness are at least three reasons in favour of this approach. A general modeling criterion is outlined and then applied to a specific case study.

1. INTRODUCTION

System simulation means the act of representing a system by a symbolic model, and using this model to produce a story of the states that can be assumed as the story of the states of the real system. An event denotes a change in the state of the simulated system. Discrete event digital simulation concerns the modeling on a digital computer of a system in which state changes can be represented by a collection of discrete events [1]. System mo-

* Istituto di Scienze della Informazione, Universita di Pisa, Pisa, Italy.
** Istituto di Elaborazione della Informazione, C.N.R., Pisa, Italy.
*** Polytechnical Institute of Kaunas, Lithuanian Republic, USSR

del changes occur when an event occurs. All modern computer simulation models use the next event approach to time advance. After all state changes have been made at the time corresponding to a particural event, simulated time is advanced to the time of the next event, where required changes are again made.

The core of any computer simulation model is then that part of the model that describes the state change rules. This description has traditionally been made either by use of program flow charts or of programming languages. In contrast to these, we are in favour of a mathematical style of description to be used at an intermediate level of model development for at least three reasons: 1) understandability of the model, 2) machine generation of simulation code, and 3) proof of correctness.

As far as the first reason is concerned, flow charts are, by their own nature, too generical and concise to give a precise understanding of what the simulation model actually does. The usefulness of a flow chart arises from the fact that it provides a generical logical description of the entire model by use of natural language assertions. On the other hand, flow charts are not suited to provide detailed descriptions, a feature that is proper of programming languages. Programming languages, however, especially simulation languages like GPSS, SIMULA, SIMSCRIPT, require a previous knowledge of the language in order to understand the model. For these reasons, the work of constructing a complete simulation model is presently very hard to divide between many persons. The man who conceived the model is, in many cases, the only one who is able to write the program that implements the model. There is a lack of general ways to communicate the intendments about the model to a programmer, unless this has

already been initiated to simulation problems

The availability of general ways of model description could also open the way to the production of software aids to simulation program building. The best way for a system analyst to study a system would be to express the simulation model of the system in general mathematical terms [2,3] and then input this description a compiler in order to produce the code of the simulation program.

Finally, as far as the proof of correctness is concerned, just remember what Goldstine and Van Neuman [4] noted years ago. A proof of program correctness can be achieved if the programmer can provide or assert a description of the state of the vector of program variables after each step, or possibly after selected steps of the program. In our case, the system state description could be used to verify the program by determining whether the (guessed) state information after the occurrence of event i is implied by the state information follwing event i-1 and the transformation performed by the event i.

2. SYSTEM DESCRIPTION

The variables that appear in a system model are used to relate one component to another and may be conveniely classified as exogenous variables, status variables, and endogenous variables [5].

Exogenous variables are the input variables of the model and are assumed to have been given independently of the system being modeled. These variables may be used in two different ways in simulation experiments. They may either be treated as given parameters which have to be read into the computer as input data or, if they are stochastic variables, they may be generated internally by the computer by known methods.

Status variables describe the state of the system or of one of its components either at the beginning of a time period, at the end of a time period, or during a time period These variables interact with both the exogenous and the endogenous variables according to the assumed functional relationships of the system.

Endogenous variables are the dependent or output variables of the system and are generated from interaction of system's exogenous and status variables according to the system's operating characteristics.

Finally, in a simulation model two other types of variables are used: the event variables and the clock. The event variables hold the time occurrence of the events that are defined in the model, while the clock holds the current simulated time.

The specification of a simulation model involves the following steps, that we shall apply next to our case study:

1) Definition of the stochastic process to be studied
2) Definition of parameters and operating characteristics
3) Definition of event variables
4) Definition of event-related state variables
5) Definition of internally generated exogenous variables
6) Definition of the rules for changing the coordinates of the process.
7) Definition of rules for computing endogenous variable values.

Step 1 is concerned with the definition of a vector of state variables, or stochastic process coordinates (e g. length of queues, state of the server etc) whose story the analyst wants to follow in order to obtain the information he is interested in In Step 2 the exogenous variables are defined. Operating characteristics for stochastic processes take the form of probability density functions (e.g. interarrival-time d f , services-time

d.f., etc.), and in general the parameters are their moments[5]. Sometimes the operating characteristics include mathematical equations relating the system's endogenous and status variables to its exogenous variables.

In Step 3 the change of states (events) of the system components are defined which are considered to be relevant for the purpose of the study. The choice of the events is strictly related to the coordinates of the stochastic process. For example, if we seek the behaviour of the state variable "number of jobs in the shop at given moment of time" then a state change occurs every time a job arrives and every time a job departs. Arrivals and departures are then system events.

In Step 4 some additional state variables can be defined which require the knowledge of the naming of the event variables and thus could not be defined in advance. Theseare the clock variable , some incremental variables (e g. no. of jobs arrived up to the i-th event) to be used as index variables at Step 5 and 6.

Step 5 defines the exogenous variables to be generated internally by the computer to simulate the system operating characteristics.

Step 6 is the most important step for the description of the system model. Here the behaviour of the system is simulated by giving assignment rules that change the value of the state variables each time an event takes place. This description combines the features ofthe detail and understandability.

Step 7 contains specification for the collection of data during the simulation run in order to compute statistical results. In this paper we shall not deal with this step further

In the following sections, Steps 1 through 6 will be described in more detail for a case study.

3. CASE STUDY

The system we have chosen is a single server queueing system with setup classes. The model is depicted in Fig.1. There are c queues or setup classes q_1, q_2 , q_c and a server S which processes jobs in a FIFO order within each class, and moves to another class as soon as the incumbent class is exhausted. The criterion according to which next queue to be served is chosen is called the server setup discipline. The objective of the simulation study is an investigation about the distribution of the number of jobs in each class and of the server busy period as far as the influence of the setup discipline and the distribution of arrival, holding and setup times is concerned. By holding time we mean the time a job takes in S, while by setup time the time to move from a given queue to another. According to these general specifications, the following model description arises, as by steps 1 through 6 above. From this model a FORTRAN IV code simulation program was then derived which yielded the results we were interested in. The purpose of this paper however is not to discuss the results but to introduce the algebraic description of the model.

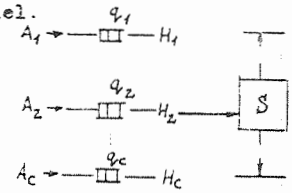

Fig. 1. The system modeled

3.1. Defitinion of the stochastic process

Let us denote by P(t) the process to be studied. According to our simulation objectives it can be defined as:

$P(t) = \{n_1(t), n_2(t), \ldots, n_c(t), \ell(t), k(t)\}$;

$n_i(t)$ = no. of users waiting in q_i at time t ;

$\ell(t)$ $\begin{cases} = 0 \text{ , if the system is idle (empty);} \\ \neq 0 \text{ , if the server S is busy in a holding or in a setup operation. Namely:} \end{cases}$

(i) when S is busy in a holding operation ($k(t) = 0$), $\ell(t)$ points to the queue S is presently attached to.

(ii) When S is busy in a setup operation ($k(t) \neq 0$), $\ell(t)$ points to the queue S is presently leaving from.

$k(t)$ $\begin{cases} = 0 \text{ , if S is presently busy in a holding operation;} \\ \neq 0 \text{ , if S is presently idle or is busy in a setup opeartion. Namely:} \end{cases}$

(j) when S is idle ($\ell(t)=0$), $k(t)$ points to the queue that was most recently served by S, i.e. the queue S is presently attached to.

(jj) When S is busy in a setup operation ($\ell(t)\neq 0$), $k(t)$ points to the queue where S will reach at the end of the setup according to the setup discipline.

3.2. Definition of exogenous variables

System operating characteristics and parameters have to be defined at this point. The former are denoted by:

A_i ($i = 1, \ldots, c$) - distribution of the interarrival time to q_i.

H_i ($i = 1, \ldots, c$) - " " " holding time in q_i

$S_{ij}(i,j=1,\dots,c)$ - distribution of the setup from q_i to q_j.

and their parameters will be:

$E(t_{A_i})$, $\sigma^2(t_{A_i})$ - interarrival time to q_i mean and variance.

$E(t_{H_i})$, $\sigma^2(t_{H_i})$ - holding time in q_i mean and variance.

$E(t_{S_{ij}})$, $\sigma^2(t_{S_{ij}})$ - setup time from q_i to q_j mean and variance.

Another parameter that will be input to the system is the setup discipline D. This can be introduced in the form of a proposition that specifies the criterion according to which server chooses next queue whenever it decides to leave the incumbent one.

3 3. Definition of event variables

Here the event list and the time advancement rules are specified. We introduce events of 4 types:

1) Type-1 events: arrivals. There exist c events of such a type because of the existence of c independent arrival sources

2) Type-2 event: end-of-holding. There exists only 1 event of such a type because we have only one server.

3) Type-3 event: end-of-setup. For the same reason as before, there exists only one an event of this type.

4) Type-4 event: end-of simulation. When this event occurs, the simulation experiment is concluded.

The event list may then be represented by the set

$$E = \{ \underbrace{e_1, e_2, \dots, e_c}_{1st}, \underset{\downarrow}{e_{c+1}}, \underset{\downarrow}{e_{c+2}}, \underset{\downarrow}{e_{c+3}} \}$$
$$\qquad\qquad\qquad\qquad\qquad\quad 2nd \quad 3rd \quad 4th$$

During the simulation run, an occurrence time is associated to each of the events above. Namely, we shall denote by $V_1(j)$ the time variable associated to event e_j $j=1,\dots,c$, by V_2 the time associated to e_{c+1}, by V_3 the time associated to e_{c+2}

and by T the time associated to e_{c+3} , the duration of the simulation run.

The advancement of the simulation time is made according to the mechanism known as "next event time advancement". This means that the simulation program scans the E-list periodically and each time chooses the event $e \in E$ whose occurrence time V is the closest to the present value of the simulated time, and not more than this.

3.4. Event dependent state variables

Having defined the event list, some additional state variables can be introduced The first is a variable we call clock holding the current value of the simulated time

It is updated according to rules that we shall see next We shall denote by m the number of events that have taken place up to the current simulated time. Accordingly,

clock (m) will denote the value of clock after the m-th occurrence of an event (clock(m) $\in$ (0,T)).

Let us write clock $(m) = t_m$. Symbol $X(t_m)$ will then denote the value of the generical state variable X at the time t_m . We shall call this the value of X after the m-th occurrence of an event and denote it by $X(m)$. Accordingly,

$V_1(j,m)$, $V_2(m)$, $V_3(m)$ are the values of $V_1(j)$, V_2, V_3 after the m-th occurrence of an event;

$n_i(m)$ - the number of users waiting in q_i after the m-th occurrence of an event;

$l(m)$ - the value of l after the m-th occurrence of an event;

$k(m)$ - the value of k after the m-th occurrence of an event.

Other variables that will be of interest are the following:

$\Gamma_1(j,m)$ = total number of users arrived up to time clock (m) into queue q_j;

$\Gamma_2(j,m)$ = total number of users that received holding service in queue q_j up to time clock (m);

$\Gamma_3(i,j,m)$ = total number of setup operations from q_i to q_j that occurred up to time clock(m).

3.5. Definition of internally generated exogenous variables

Rules are now to be given for the simulation of the operating characteristics A_i, H_i, S_{ij} defined above. To this purpose we introduce the sequences of pseudorandom numbers:

$\{\xi_i^{(j)}\} = \{\xi_1^{(j)}, \xi_2^{(j)}, \dots\}$ $j=1,2,\dots,c$, sequence of interarrival time values generated according to the A_j distribution;

$\{\eta_i^{(j)}\} = \{\eta_1^{(j)}, \eta_2^{(j)}, \dots\}$ $j=1,2,\dots,c$, sequence of holding time values generated according to the H_j distribution;

$\{\zeta_i^{(ij)}\} = \{\zeta_1^{(ij)}, \zeta_2^{(ij)}, \dots\}$ $ij=1,2,\dots,c$ sequence of setup time values generated according to the S_{ij} distribution.

The values of those sequences are generated according to Monte Carlo methods like the linear congruential inverse transform method, etc. [1, 5] i.e. each value is obtained from the preceding by known rules that we simply denote by

$$\xi_k^{(j)} = f\left[\xi_{k-1}^{(j)}, E(t_{A_j}), \sigma^2(t_{A_j})\right]$$

and similarly for the η and ζ values.

3.6. Rules for changing the coordinates of the process

Let us denote by $e(m)$ the m-th event selected. The next

event time advancement criterion can be expressed as follows:

$$e(m+1) = e \in E \mid V = \min\{\|V_4(j,m)\|, V_2(m), V_3(m), T\}$$
$$\forall\ V_k \geq clock(m),\ k = 1,2,3,$$

where V denotes the occurrence time of the generical event $e \in E$. According as the selected event is of the 1-st, 2nd, 3rd or 4th type, one of the following sets of assignment rules is chosen. Each set involves a different sequence of assignments to the state variables defined above.

3.6.1. Case of an event of the 1st-type $(e(m+1) \in \{e_1, \dots, e_c\})$.

In this case are faced with an arrival event. Let us assume that the arrival is on queue number p, that is $\min\{\dots\} = V_1(p,m)$. The simulation program will then enter an event routine which will make the following changes in sequence:

$n_j(m+1) = n_j(m)$, for $j = 1, \dots, c$, $j \neq p$,

$n_p(m+1) = n_p(m+1)$,

$r_1(j, m+1) = r_1(j,m)$, for $j = 1, \dots, c$, $j \neq p$,

$r_1(p, m+1) = r_1(p,m) + 1$,

$\|r_2(j, m+1)\| = \|r_2(j,m)\|$,

$\|r_3(i, j, m+1)\| = \|r_3(i,j,m)\|$,

$$\ell(m+1) = \begin{cases} \ell(m), & \text{if } \ell(m) \neq 0, \\ p, & \text{if } [(\ell(m) = 0)] \wedge [k(m) = p], \\ k(m), & \text{if } [\ell(m) = 0] \wedge [k(m) \neq p], \end{cases}$$

$$k(m+1) = \begin{cases} k(m), & \text{if } \ell(m) \neq 0, \\ 0, & \text{if } [\ell(m) = 0] \wedge [k(m) = p], \\ p, & \text{if } [\ell(m) = 0] \wedge [k(m) \neq p], \end{cases}$$

$clock(m+1) = V_1(p,m)$,

$V_4(j, m+1) = V_4(j,m)$, $j = 1, 2, \dots, c$, $j \neq p$,

$V_1(p, m+1) = clock(m+1) + \xi^{p}_{r_1(p,m+1)}$

$$V_2(m+1) = \begin{cases} clock(m+1) + \eta^{p}_{r_1(p,m)+1}, & \text{if } [\ell(m) = 0] \wedge [k(m) = p], \\ V_2(m), & \text{otherwise} \end{cases}$$

In the first case V_2 represents the end of the holding time of the $r_2(p, m+1)$ - th user. In the second case this time cannot be forecast and then we set it to an infinite value.

$$V_3(m+1) = \begin{cases} \underline{\text{clock}}\,(m+1) + \sum_{r_3(k(m),p,m)+1}^{(k(m),p)} & , \text{if } [\ell(m)=0] \wedge [k(m) \neq p] \\ V_3(m) & , \text{otherwise} \end{cases}$$

Similarly, in the first of the two cases, V_3 represents the end time of the $r_3(k(m), p, m+1)$ -th setup operation.

3.6.2. Case of an event of the 2nd-type $(e(m+1) = e_{c+1})$

In this case we are faced with an end of holding time event. That is $\min\{\dots\} = V_2(m)$ and $\ell(m)$ is the number of the queue on which the event has taken place. The corresponding routine will have to do the sequence:

$n_j(m+1) = n_j(m)$, $\quad j = 1, \dots, c$, $j \neq \ell(m)$,

$n_{\ell(m)}(m+1) = n_{\ell(m)}(m-1)$,

$\| r_1(j, m+1) \| = \| r_1(j, m) \|$,

$r_2(j, m+1) = r_2(j, m)$, $\quad j = 1, \dots, c$, $j \neq \ell(m)$,

$r_2(\ell(m), m+1) = r_2(\ell(m), m) + 1$,

$\| r_3(i, j, m+1) \| = \| r_3(i, j, m) \|$,

$\ell(m+1) = f_1(D)$,

$k(m+1) = f_2(D)$.

where D is the setup discipline.

$\underline{clock}(m+1) = V_2(m)$,

$V_1(j, m+1) = V_1(j, m) \qquad j = 1, 2, \dots, c$,

$$V_2(m+1) = \begin{cases} clock(m+1) + \eta_{r_2(\ell(m),m)+1}^{\ell(m+1)} & , \text{ if } [\ell(m+1) \neq 0] \wedge [k(m+1) = 0], \\ \infty & , \text{ otherwise.} \end{cases}$$

In the first case V_2 provides the end time of the next holding operation on the current queue. In the second case, V_2 is set to ∞ because it is not possible to forecast this time when queue S is presently attached to is empty.

$$V_3(m+1) = \begin{cases} clock(m+1) + \Xi_{r_3(\ell(m),k(m),m)+1}^{\ell(m),k(m+1)} & , \text{ if } [\ell(m+1) \neq 0] \wedge [k(m+1) \neq 0], \\ \infty & , \text{ otherwise .} \end{cases}$$

Similarly V_3 provides the end of the next setup only in case the queue S is presently serving is empty.

3.6.3. Case of an event of the 3rd-type ($e(m+1) = e_{c+2}$).

In this case we have to face with an end-of-setup event. That is $\min\{\dots\} = V_3(m)$ and $\ell(m), k(m)$ provide the number of initial and final queue of the setup operation.

The appropriate event routine will have to do the following operations, in sequence:

$n_j(m+1) = n_j(m)$, $\quad j = 1, \dots, c$,

$\| r_1(j, m+1) \| = \| r_1(j, m) \|$,

$\| r_2(j, m+1) \| = \| r_2(j, m) \|$,

$r_3(i, j, m+1) = r_3(i, j, m) \qquad (i \neq \ell(m),\ j \neq k(m))$,

$r_3(\ell(m), k(m), m+1) = r_3(\ell(m), k(m), m) - 1$,

$\ell(m+1) = k(m)$,

$k(m+1) = 0$

$$\underline{\text{clock}}\,(m+1) = V_3(m),$$

$$V_1(j, m+1) = V_1(j, m), \quad j = 1, \ldots, c,$$

$$V_2(m+1) = \text{clock}\,(m+1) + \eta^{\ell(m+1)}_{r_2(\ell(m+1),\, m)+1},$$

$$V_3(m+1) = \infty.$$

V_2 provides the end time of the holding that begins on queue $\ell(m+1)$ that S has reached because of the setup we have simulated.

3.6.4. Case of an event of 4th-type ($e(m+1) = e_{c+3}$).

This section of the simulation program is entered at the end of the simulation run. It is supposed to make statistical evaluation on data that were collected during the course of the simulation, and to print the results. Namely it will provide estimations about the stochastic process P(t). For the sake of conciseness we did not represent the operation for data collection. It is however clear that this should be done periodically during the course of the simulation according to known sampling techniques [1].

4. CONCLUSIONS

A method has been introduced for the description of a simulation model at an intermediate level of its development. The description places itself midway between the level of a general flow chart and that of the coded simulation program level. This approach improves the model by combining the festures of the detail and understandability, two prerequisites for the proof of correctness and the machine generation of the simulation code.

REFERENCES

1. FISHMAN, G.S.: "Concepts and Methods in Discrete Event Digital Simulation", Wiley, 1973.
2. ПРАНЕВИЧЮС Г.И. Применение метода управляющих последовательностей при моделировании сложных систем, Математика и математическое моделирование, Вып.I, Каунас, 1976, с. 32-
3. ПРАНЕВИЧЮС Г.И. Об одном методе моделирования сложных систем, Математическое обеспечение сложных систем, ч I, Киев, 1977, с. I2-I5.
4. GOLDSTINE, H.H., J.Von NEUMAN: "Planning and Coding Problems for an Electronic Computer Instrument", Part 2, Voll. 1-3; In "John Von Neuman Collected Works", Vol. 5, A.H. Traub Editor, Pergamon, 1963, 80-235.
5. NAYLOR, T.H., BALINTFY, J.L., BUDRICK, D.S., CHU, K.: "Computer Simulation Techniques", Wiley, 1966.