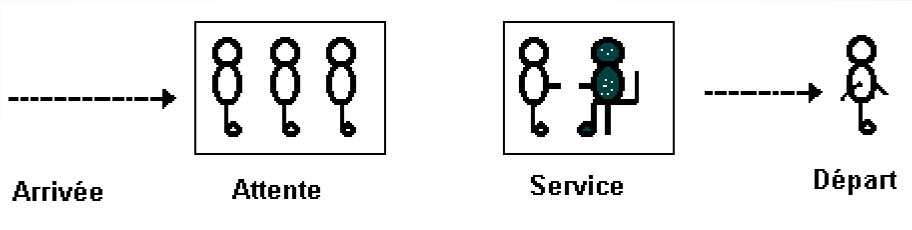
University of Msila
FACULTY OF MATHEMATICS AND INFORMATICS
DEPARTMENT OF COMPUTER SCIENCE

Modeling of dynamic discret events systems 2

(DDES)

SAED modeling approaches

- **EVENT-BASED APPROACH**
  In this approach, a system is modeled by defining the changes that take place when events occur. The designer must therefore determine which events can change the state of the system, and then develop the logic associated with each type of event.
  Example:
  The system to be modeled is a bank where customers arrive and are served by a single waiter (employee). Customers arrive at the bank, wait in front of a counter, are served and then leave the bank.



- he state of the system is defined by the state of the **server and the number of waiting customers**. In this way, the state remains constant except when a customer arrives or when a customer leaves the bank.
- The event-based approach describes what happens when a customer arrives and when a customer leaves the service. Since the system only changes state
  at the instants of these two events (customer arrival or customer departure), these are sufficient to fully describe the dynamics of the system.

SAED modeling approaches

Evénement : **Arrivée_Client**

Planifier ou prévoir la prochaine Arrivée ;

Si le **SERVEUR** est Occupé

Alors

/* on incrémente le nombre de clients en attente */

- **Clients_En_Attente** ← **Clients_En_Attente + 1**

Sinon

*/*le serveur est libre, Changer son état à Occupé */*

- **ETAT_SERVEUR** ← Occupé
- Planifier un Evénement **Fin_De_Service** pour TNOW + Durée de service du client

Fin

Fin événement Arrivée_Client

SAED modeling approaches

<u>Evénement :</u> **Fin_De_Service**

Ejecter le client du système ;
<u>Si</u> **Clients_En_Attente** > 0
<u>Alors</u>
/* on décrémente le nombre de clients en attente */
/* on engage un nouveau client dans le service */

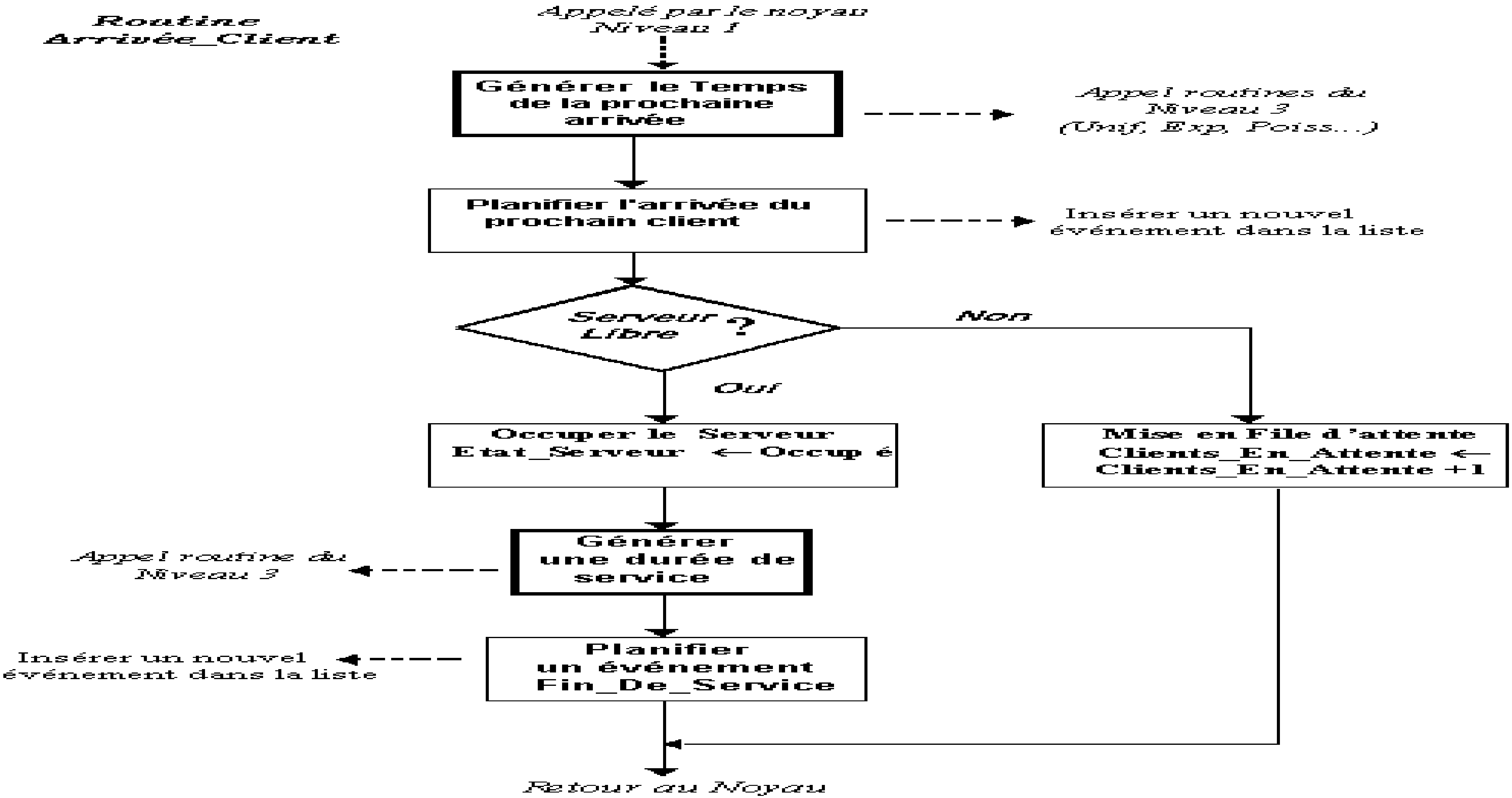
- **Clients_En_Attente** ← **Clients_En_Attente - 1**
- Planifier un Evénement **Fin_De_Service** pour ce client à la date : TNOW + Durée de service du client

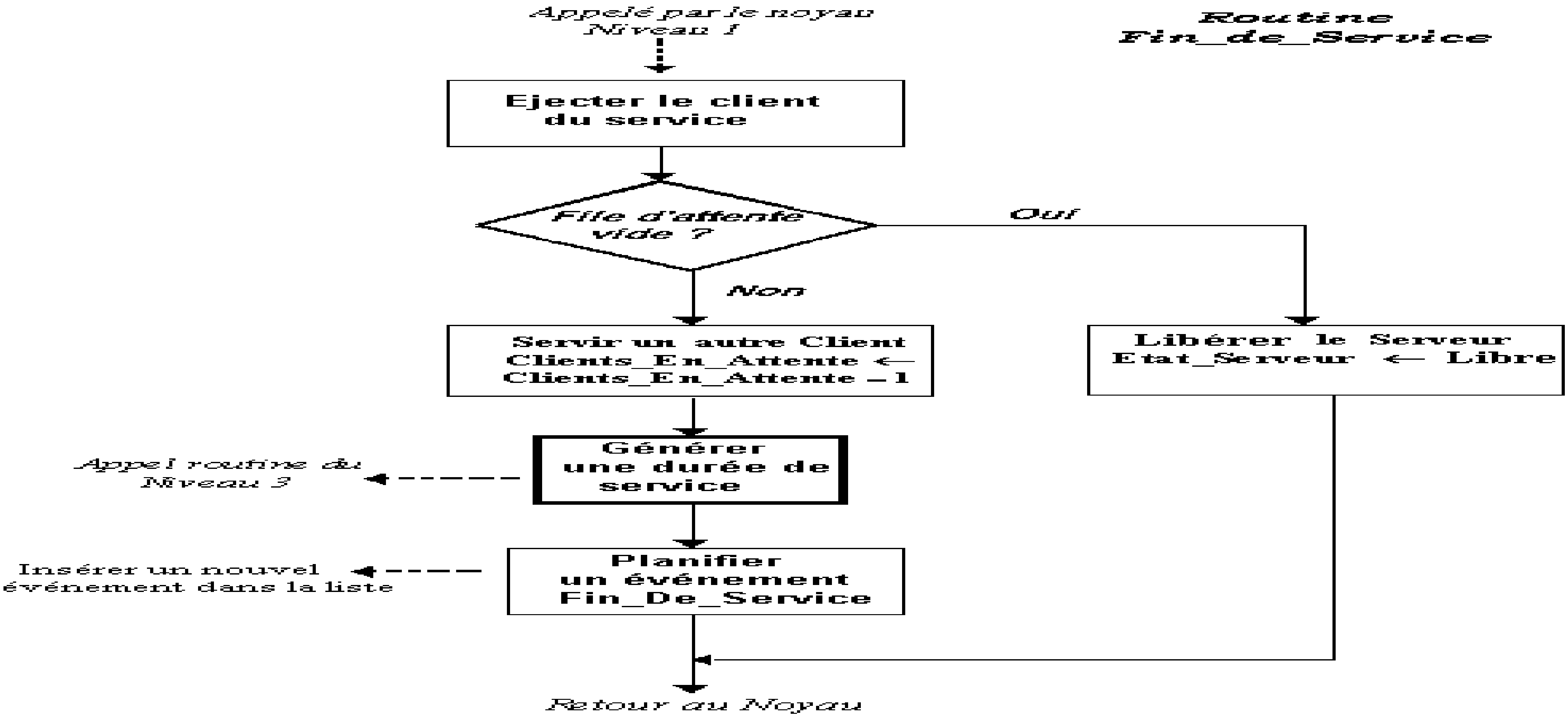
<u>Sinon</u>
/* le serveur vient de finir de servir un client et */
/* *il n' y a plus de clients en attente* */

- **ETAT_SERVEUR** ← Libre

<u>Fin</u>

<u>Fin événement</u> **Fin_De_Service**


Routine
Arrivée_Client
Appelé par le noyau
Niveau 1
Générer le Temps de la prochaine arrivée
Appel routines du Niveau 3 (Unif, Exp, Poiss...)
Planifier l'arrivée du prochain client
Insérer un nouvel événement dans la liste
Serveur Libre ?
Non
Oui
Occuper le Serveur
Etat_Serveur ← Occupé
Mise en File d'attente
Clients_En_Attente ← Clients_En_Attente +1
Générer une durée de service
Appel routine du Niveau 3
Planifier un événement Fin_De_Service
Insérer un nouvel événement dans la liste
Retour au Noyau



Routine Fin_de_Service
Appelé par le noyau Niveau 1
Ejecter le client du service
File d'attente vide ?
Oui
Non
Servir un autre Client Clients_En_Attente ← Clients_En_Attente – 1
Libérer le Serveur Etat_Serveur ← Libre
Générer une durée de service
Appel routine du Niveau 3
Planifier un événement Fin_De_Service
Insérer un nouvel événement dans la liste
Retour au Noyau

SAED modeling approaches

- In the event-driven approach, the kernel's main tasks are :
  1) **Time control:** determine the date of the next event and initialize the clock with this date
  2) **Identify events:** determine which events must take place at the current time (clock)
  3) **Execute events:** trigger the events identified as due to take place.

Example:

The arrival of a customer will trigger the addition of an End_Of_Service event to the calendar. It also triggers the addition of a Customer_Arrival event corresponding to the next customer.

The end of a service can trigger the addition of an End_Of_Service event to the schedule, if the queue is not empty, and which in this case concerns the new customer in the queue.

In this way, each event in the list must include at least the following two items of information:

- the date of occurrence of the event
- the event identification (usually a number)

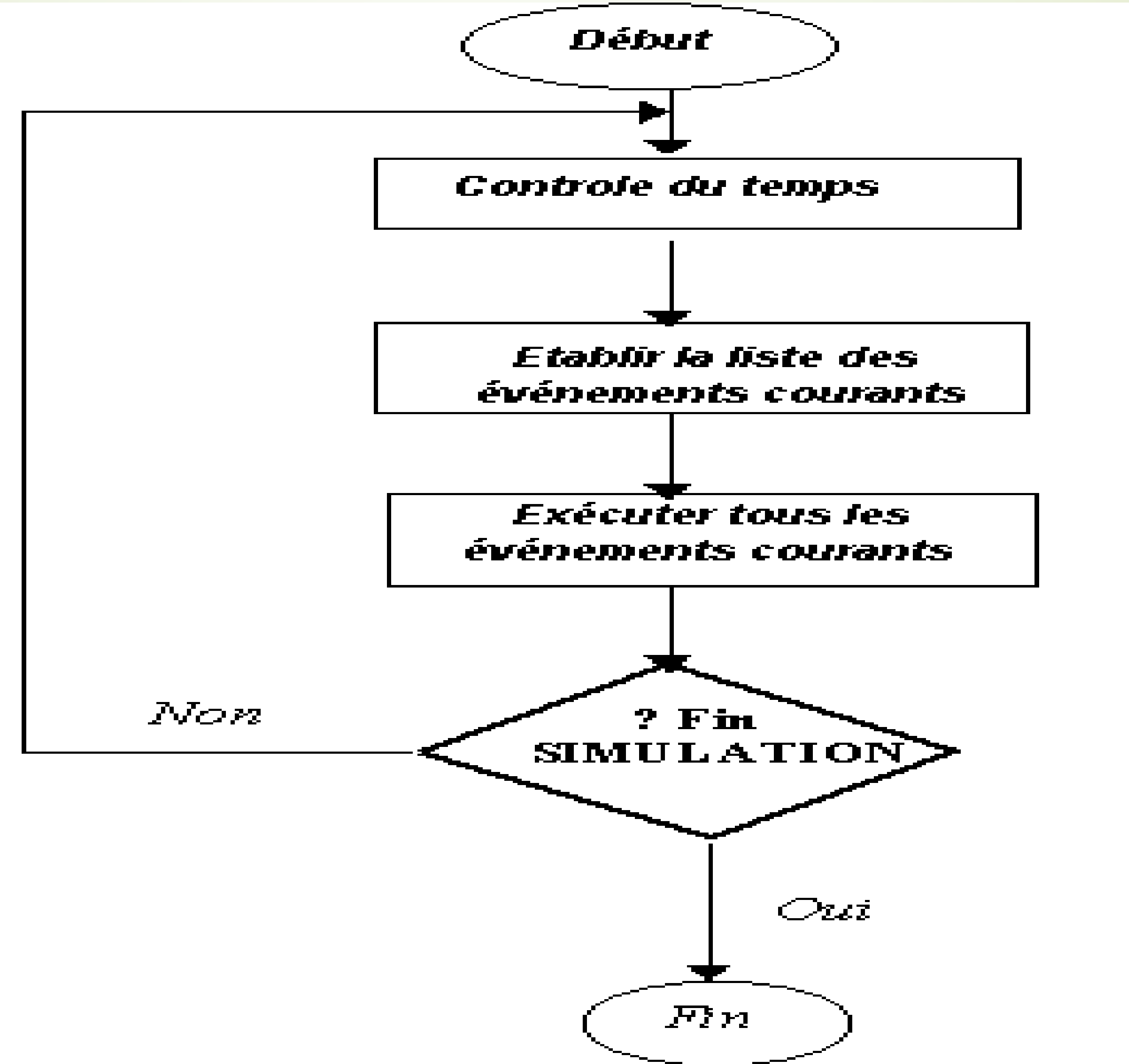
Information on the entities involved in the event may also be useful (e.g.: selection of a customer in the queue).

As the simulation progresses, the kernel will execute a 2-phase cycle:

SAED modeling approaches

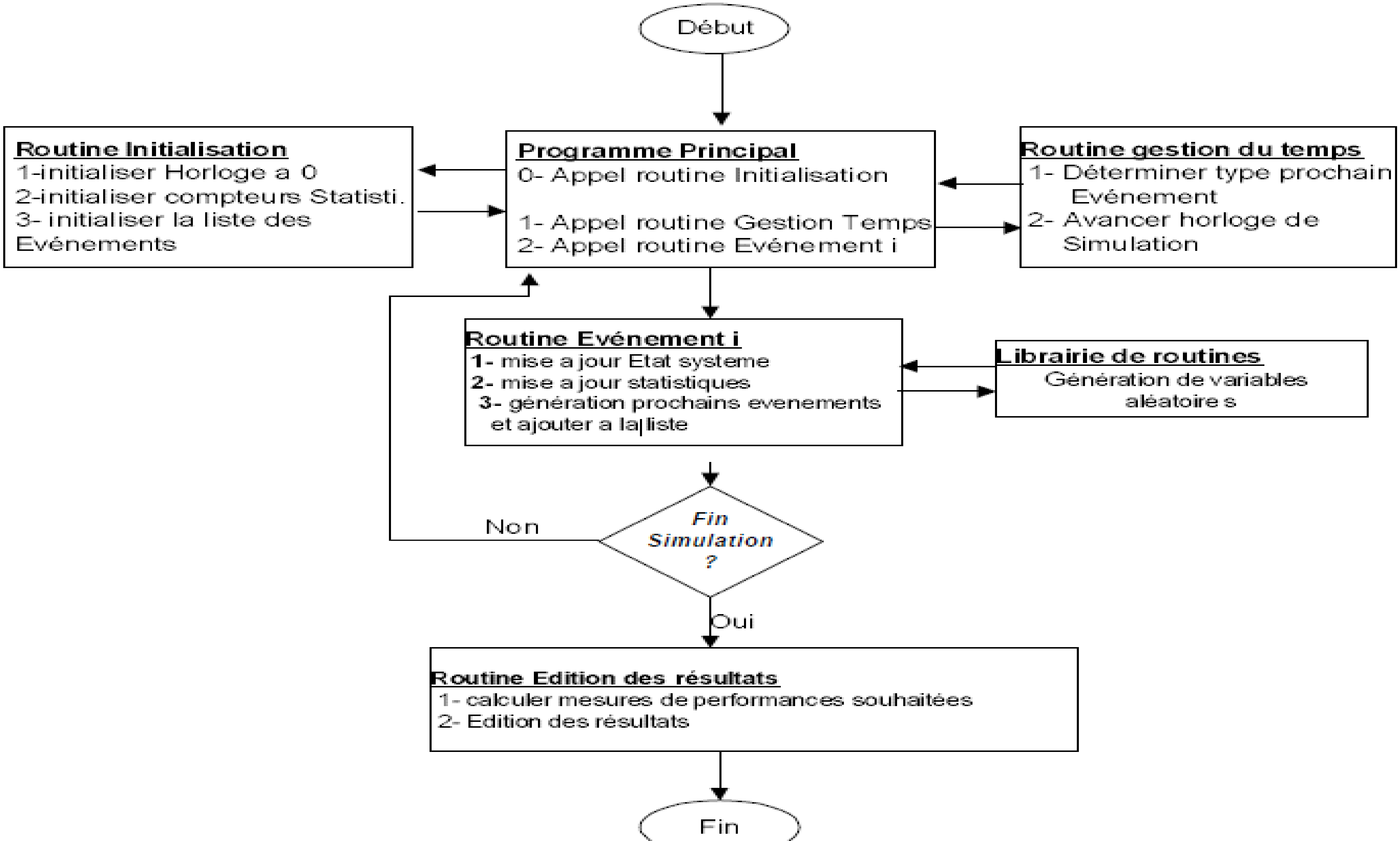
- 1) **Time control: this phase includes**
  a) determining the date of the next event by examining the calendar (list of events)
  b) initializing the clock with the date of the next event
  c) construction of a list of current events including all events whose date of occurrence is equal to the clock.
- 2) **Execution of current events**
  Current events are executed under kernel control. No event can be triggered without the kernel. This ensures that the logical sequence of events is entirely controlled by the kernel. Once an event has been executed, it is deleted (from the calendar and from the list of current events).
  This cycle is repeated until the end of the simulation. A simplified kernel flowchart might look like this:

SAED modeling approaches



SAED modeling approaches

- **Components of a discrete-event simulation model**
  The following components are found in most discrete-event simulation models adopting the "event" approach:
- **System state**: defined by a collection of state variables describing the state of the system at a particular time.
- **Simulation clock**: a variable indicating the current value of simulation time.
- **Event list**: A list containing the dates of occurrence of events scheduled to take place in the future.
- **Statistical counters:** Variables used to collect statistical information on system performance.
- **Initialization routine.** Subroutine used to initialize the simulation model at time zero.
- **Time management routine** (time control): subroutine which determines the next event from the event list and initializes the simulation clock with the date of occurrence of this current event.
- **Event routines:** subroutines that update the system state when a particular type of event occurs (there is an event routine for each type of event).
- **utility routine library:** set of subroutines used to generate random variables identified as part of the simulation model.
- **Results generator:** subroutine which calculates estimates (from statistical counters) of the desired performance measures and generates a report at the end of the simulation.
- **Main program:** responsible for initializing the system state at the start of the simulation, and all variables used during the simulation. It invokes the time management routine to determine the next event to take place and passes control to the routine associated with that event. It also checks whether the end of the simulation has been reached and, if so, invokes the results generator.


Début
Routine Initialisation
1-initialiser Horloge a 0
2-initialiser compteurs Statisti.
3- initialiser la liste des Evénements
Programme Principal
0- Appel routine Initialisation
1- Appel routine Gestion Temps
2- Appel routine Evénement i
Routine gestion du temps
1- Déterminer type prochain Evénement
2- Avancer horloge de Simulation
Routine Evénement i
1- mise a jour Etat systeme
2- mise a jour statistiques
3- génération prochains evenements et ajouter a la liste
Librairie de routines
Génération de variables aléatoire s
Non
Fin Simulation ?
Oui
Routine Edition des résultats
1- calculer mesures de performances souhaitées
2- Edition des résultats
Fin

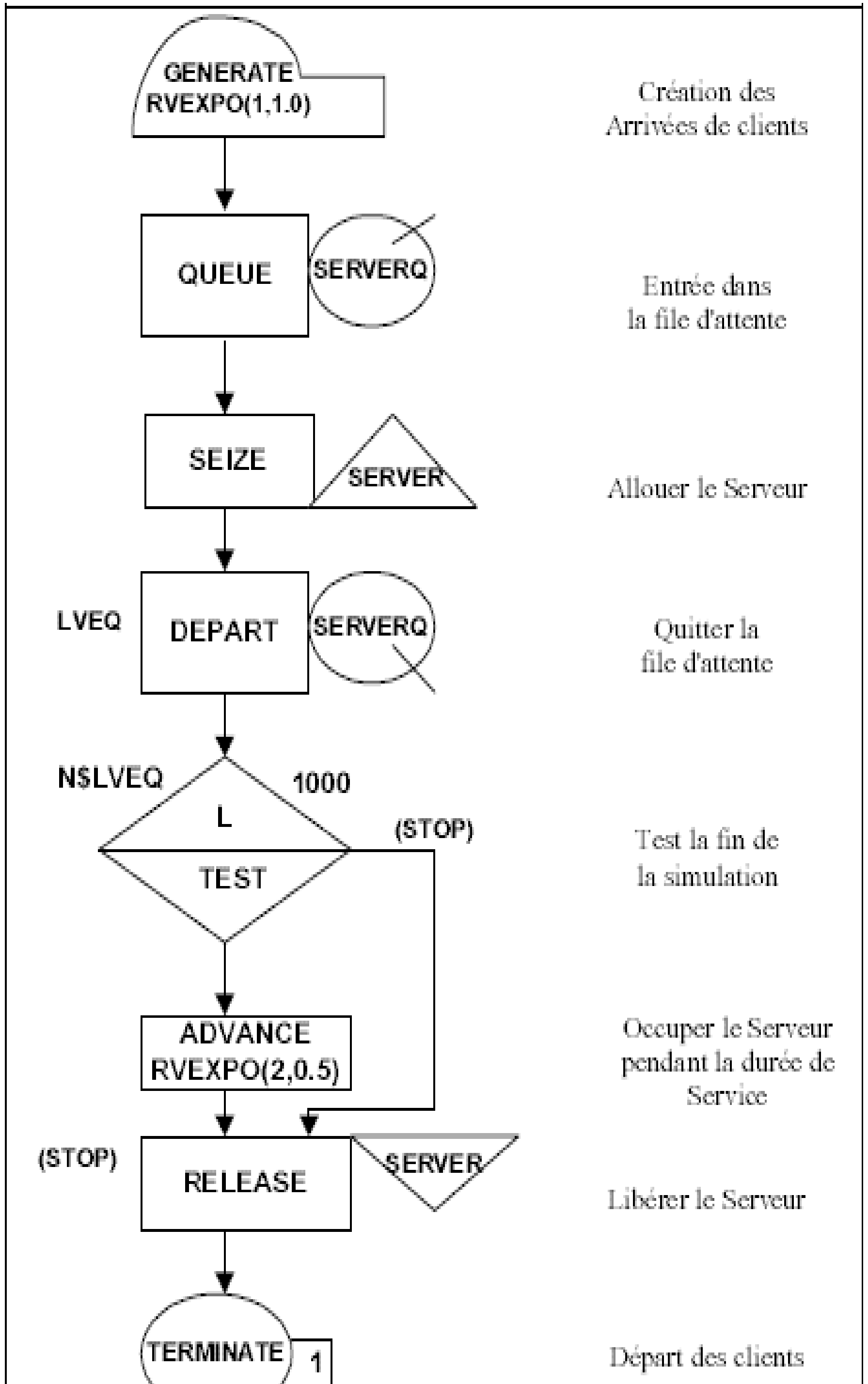
Simulation languages

- The first simulation languages appeared around 1960, and were more oriented towards the representation and simulation of discontinuous (discrete-event) systems.

Simulateur	Pays	Date
SIMSCRIPT	USA	1963
SIMULA	Norvège	1966
GPSS	USA	1968
GASP	USA	1974
Q-GERT	USA	1977
SLAM	USA	1979
QNAP	France	1980
CAPS/ECSL	Grande Bretagne	1980
SIMAN	USA	1982

Simulation languages

GPSS (General Purpose Simulation System) is a language offering a process-oriented approach. It was first developed by IBM in 1961, and several versions have since been released, the most recent of which is GPSS/V. GPSS is one of the forerunners of the "process" modeling approach, and has graphical representation support like most current languages.




```
*
*       Exemple de  Modèle en GPSS
*
        SIMULATE
        GENERATE    RVEXPO(1,1.0)      creation -Injection des entités dans le système
        QUEUE       QSERVER            entrée dans la file d'attente
        SEIZE       SERVER             demande d'allocation du  serveur
LVEQ    DEPART      QSERVER            sortie de la la file d'attente
        TEST L      N$LVEQ,1000,STOP   Test si l'execution est arrivée a sa fin
        ADVANCE     RVEXPO(2,0.5)      occuper le serveur pendant t = durée de service
STOP    RELEASE     SERVER             libérer le serveur
        TERMINATE   1                  entité quitte le système
*
*       Instructions    de Controle
*
        START       1000               faire 1 seule simulation qui dure 1000 unités
        END
```

RELATIVE CLOCK: 1014.1565 ABSOLUTE CLOCK: 1014.1565

BLOCK CURRENT	TOTAL
1	1000
2	1000
3	1000
LVEQ	1000
5	1000
6	999
STOP	1000
8	1000

--AVG-UTIL-DURING --

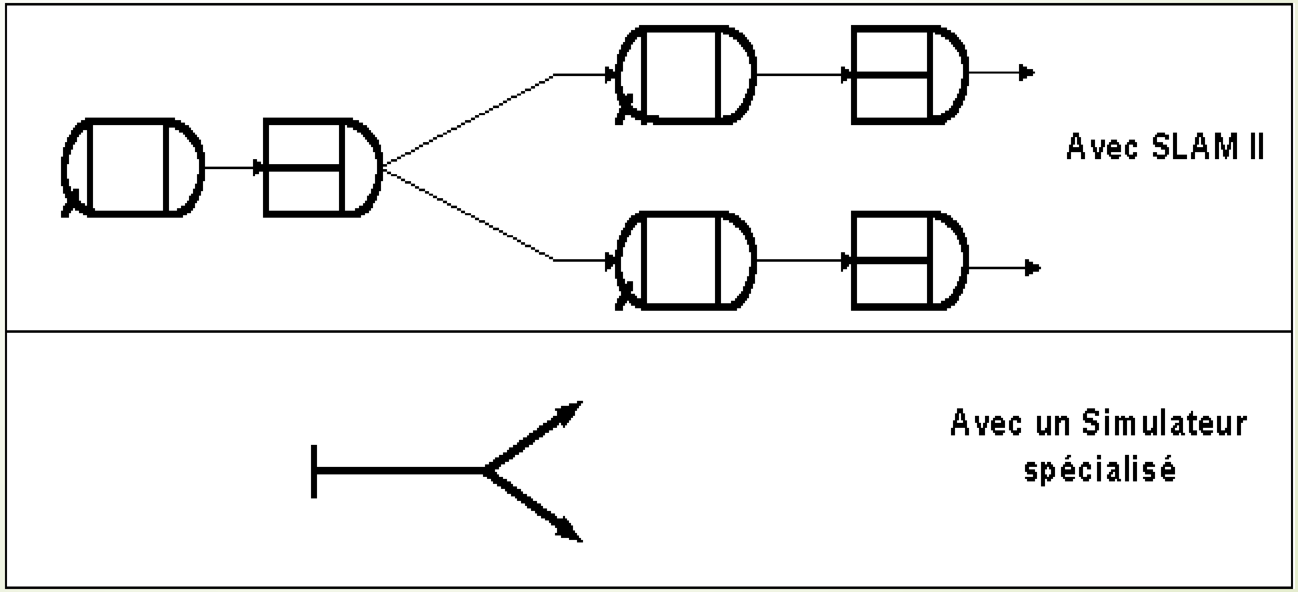
FACILITY	TOTAL TIME	AVAIL TIME	UNAVL TIME	ENTRIES	AVERAGE TIME/XACT	CURRENT STATUS	PERCENT AVAIL	SEIZING XACT	PREEMPTING XACT
SERVER	0.516			1000	0.523	AVAIL			

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/UNIT	$AVERAGE TIME/UNIT	QTABLE NUMBER	CURRENT CONTENTS
SERVERQ	8	0.605	1000	454	45.4	0.614	1.124		0

RAMDOM STREAM	ANTITHETIC VARIATES	INITIAL POSITION	CURRENT POSITION	SAMPLE COUNT	CHI-SQUARE UNIFORMITY
1	OFF	100000	101001	1001	0.71
2	OFF	200000	200999	999	0.69

Specialized simulators

- A specialized simulator is a simulation tool offering a description language (or an interactive data input system) in which the instructions (or primitives) are objects of the system to be modeled. The primitives offered are, in this case, aggregates of elementary processes (queues, resources, activities, etc ...) representing a particular object of the system (machine, stock, conveyor, pallet, etc ... in the case of a production system) and its behavior.
  One of the advantages of specialized simulators is the reduced conceptualization effort required in the modeling stage. The following figure shows how easy it is to represent the same problem (here, the modeling of a diverging switch in a handling network) using a specialized "handling network" simulator, and a general simulation language (here, SLAM II).



Specialized simulators

Simulateur	Sociétés	Domaine
MAP/1	Pritsker Associates - USA	systèmes de production
SIMFLEX	INRIA/SIMULOG - France	Ateliers Flexibles
SAME/AGVS	SERI/RENAULT Automation	Circuits de manutention
SIMUFLEX	RAMSES Automation - France	Ateliers Flexibles
NETWORK II.5	CACI-Los Angeles - USA	Réseaux locaux LAN
COMNET II.5	CACI-Los Angeles - USA	Réseaux Télécom WAN