Université de Msila

Faculté de mathématique et d'informatique

Département d'informatique

Modélisation des systèmes dynamiques à événements discrets (SAED)

Introduction

➢ **Modélisation d'un système :** La modélisation consiste a définir les points suivants :

- **Le système** : existant (ou pas) auquel se réfère le modèle.
- **Le modèle** : représentation abstraite du système (simplifie)
- **L'objectif** : le but pour lequel le modèle a été élabore.
- **Un critère de rentabilité** : un critère économique que justifie l'utilisation d'un modèle

■La modélisation d'un système dynamique demande au préalable

1. la définition d'une échelle des temps

2. la définition d'une variable d'entrée **u** et une variable de sortie **y**.

- Comment définir l'entrée et la sortie d'un système?
- La règle générale est définir comme entrées toutes les variables qui peuvent être contrôlées ou modifiées et comme sorties toutes les variables qui peuvent être mesurées.

Introduction

➢ **Les méthodes de modélisation:**

Plusieurs méthodes de modélisation sont aujourd'hui utilisées. Chacune d'elles étant plus ou moins bien adaptée a des aspects spécifiques d'analyse des performances d'un système donne .Les classes de méthodes de modélisation se divisent en deux classes:

- les méthodes descriptives
- les méthodes analytiques

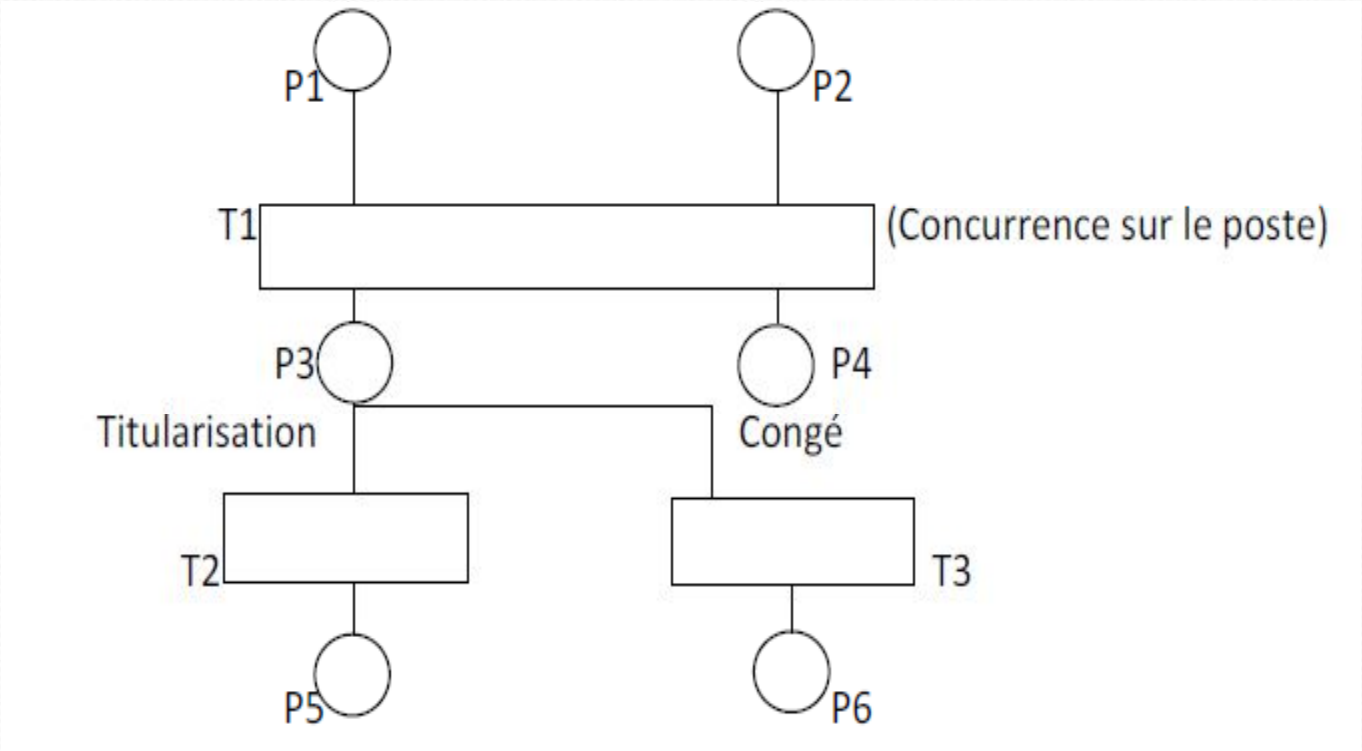
■**Les méthodes descriptives :** Elles permettent de décrire le comportement logique d'un système généralement sans faire intervenir l'aspect temporel.

Un exemple de méthodes descriptives : les réseaux de Petri. Ils fournissent un modèle graphique de description des opérations d'une machine informatique. Ils sont particulièrement adaptes pour décrire des phénomènes de concurrences, de conflits et la synchronisation.

Introduction

Exemple : Gestion du personnel par RDP

Toute personne peut demander un recrutement selon la disponibilité des postes budgétaires. Un employé peut être titulaire Un employé peut partir en congé, formation...



■ **Les méthodes analytiques :** Ce sont des méthodes basées sur des modèles mathématiques, et leurs résultats sont obtenus par calcul. De nombreuses recherches dans le domaine de la modélisation se sont focalisées sur la théorie de la file d'attente. Plusieurs modèles de file d'attente sont établis.

Exemple :

le modèle M/M/1 : Arrivée suivant la loi de Poisson

Service suivant la loi exponentielle

Un serveur

le modèle M/M/S : Même chose

S serveurs

Exemple de modélisation par files d'attente

Arrivée des clients devant un guichet de banque

Objectifs :

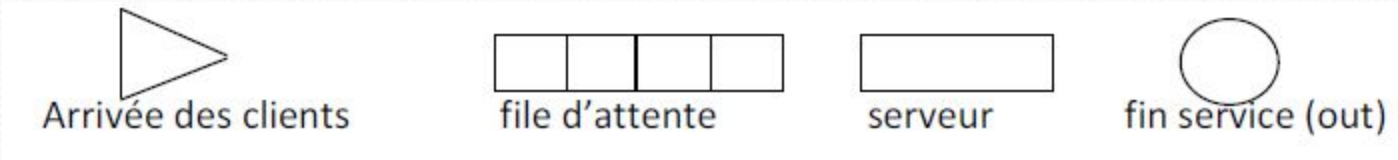
- comprendre la fonction du système
- Réduire le temps d'attente

Alternatives :

- Un ou deux guichets ?
- Un ou deux files d'attentes ?

Une méthode analytique peut avoir des aspects d'une méthode descriptive

Exemple : représentation d'une file d'attente.



Système a événements discrets

➢ Composants d'un modèle a événements discrets

Un modèle est caractérisé par un aspect statique lié à la structure du système modélisé, et par un aspect dynamique lié à l'évolution du système au cours du temps.

▪ Partie statique

Un modèle à événements discrets est composé d'objets ou entités (pièces, machines, etc ...) avec lesquelles s'effectuent au cours du temps, des services qui peuvent être actifs (activités ou opérations) ou passifs (attente), et de relations entre ces objets (possibilité pour une pièce de passer sur une machine ...). On distingue en général deux types d'objets :

- **les clients ou usagers :** Sur lesquels sont effectués les services (pièces dans un atelier, clients dans un magasin, informations dans un réseau de transmission, etc ...) ;souvent, les clients sont des objets circulants (en particulier dans les systèmes de production).

- **les ressources :** Sont nécessaires pour effectuer les services ; ces ressources peuvent être composées d'une ou plusieurs unités (nombre de machines identiques, de personnel ayant la même compétence, de chariots, de places dans une file d'attente, etc ...

Système a événements discrets

- **Partie dynamique**

L'aspect dynamique d'un modèle réfère aux mécanismes de changements d'état. Lorsque le temps évolue, des interactions se produisent entre les objets qui composent le modèle et ceux-ci changent alors d'état. Ces changements d'état peuvent se faire de façon continue, discrète, ou une combinaison des deux. C'est ainsi qu'on parle de modèles "continus", modèles à "événements discrets", et modèles "combinés événements discrets continus

CONCEPTS LIES A LA METHODE DE SIMULATION des SAED

➢VARIABLES D'ETAT D'UN SYSTEME

Les variables d'état d'un système sont toutes les informations nécessaires pour définir ou caractériser ce qui est en train de se passer dans le système à un niveau de détail suffisant et ce à un instant donné.

Exemple:

Dans une banque, le temps d'inoccupation des guichets n'a pas d'intérêt ou ne constitue pas une variable d'état du système analysé si on décide d'affecter un client au premier guichet libre et ce de manière cyclique. Par contre si on décide d'affecter un client en priorité au guichet qui est resté le plus longtemps inoccupé, alors le temps d'inoccupation des guichets devient une variable d'état du système.

- Dans un modèle à événements discrets, les variables d'état restent constantes sur des intervalles de temps et changent de valeurs uniquement à certains points bien définis appelés : **Temps ou instants d'événements.** Par contre les modèles continus ont des variable d'état définies par des équations différentielles ou aux différences qui permettent aux variables de changer de façon continue au cours du temps.

CONCEPTS LIES A LA METHODE DE SIMULATION des SAED

➢ **ENTITES, RESSOURCES, ATTRIBUTS**

■ Les entités désignent des objets du système modélisé. Le terme en lui-même peut désigner aussi bien des objets passifs qui subissent des opérations que des objets actifs qui réalisent des opérations.

• Les entités qui se déplacent dans le système (ex: clients) au cours de la simulation sont qualifiées de dynamiques alors que les entités qui ne se déplacent pas et servent simplement d'autres entités sont qualifiées de statiques (ex : serveur dans une banque).

•Une entité permanente est une entité qui reste dans le système même lorsque la simulation est terminée (ex: machine , serveur). Une entité temporaire est une entité qui subit des opérations et quitte le système dès qu'elle termine ses opérations.

■**Les ressources :** ce sont des objets qui exécutent des opérations et en général ne se déplacent pas dans le système (Machine, Opérateur, Unité centrale, ...). Cependant, il faut noter qu'il est tout a fait possible de rencontrer en pratique des objets de type ressource qui se déplacent a l'interieur du système tout en exécutant l'opération (chariot transportant une pièce dans un atelier).

• Il existe plusieurs états possibles pour une ressource. Les deux états minimum sont : **libre** ou **occupée** mais d'autres possibilités existent telles que : en panne , bloquée, en famine , ...

CONCEPTS LIES A LA METHODE DE SIMULATION des SAED

- Un objet (entité ou ressource) est caractérisé par un ou plusieurs attributs auxquels on peut affecter des valeurs. Ainsi les attributs peuvent être considérés comme des variable locales à l'objet. On distingue cependant, deux types d'attributs :

Fixes : contiennent les caractéristiques constantes de l'objet (durée de service, date d'arrivée dans le système, couleur d'une pièce, ...)

Variables : contiennent les caractéristiques changeantes de l'objet (état d'une ressource, longueur de la file associée à une ressource, temps d'usinage restant pour une entité...)

Les attributs qui sont intéressant dans une investigation peuvent ne pas l'être dans une autre. Par exemple si on a des pièces rouges et des pièces bleues à usiner, l'attribut **couleur** sera certainement un attribut intéressant. Par contre si on s'intéresse uniquement au temps de séjour dans le système par toutes les pièces, la couleur d'une pièce peut ne pas être un attribut important.

CONCEPTS LIES A LA METHODE DE SIMULATION des SAED

➢ Evénement , Activité , Processus

■Evénement

Un événement est caractérisé par une date (date d'événement) à laquelle le système change d'état. On distingue les événements internes au système et externes appelés aussi **événements endogènes et événements exogènes.**

Exemple :

- Le début de service d'un client est un événement endogène du fait qu'il est interne au système qu'on est en train de simuler.
- L'arrivée d'un client pour le service est un événement exogène du fait que ce phénomène est en dehors de la simulation. Cependant l'arrivée d'un client au service empiète sur le système et doit être pris en considération.

■ Activité

A chaque événement les objets concernés s'engagent dans des opérations. Ces opérations qui sont initiées (ou terminées) à chaque événement sont appelées des **activités.** Toute activité est limité à son début et à sa fin par un événement.

Exemple :

- Le début de service d'un client va initier les opérations : **extraire client** de la file**, servir le client**

CONCEPTS LIES A LA METHODE DE SIMULATION des SAED

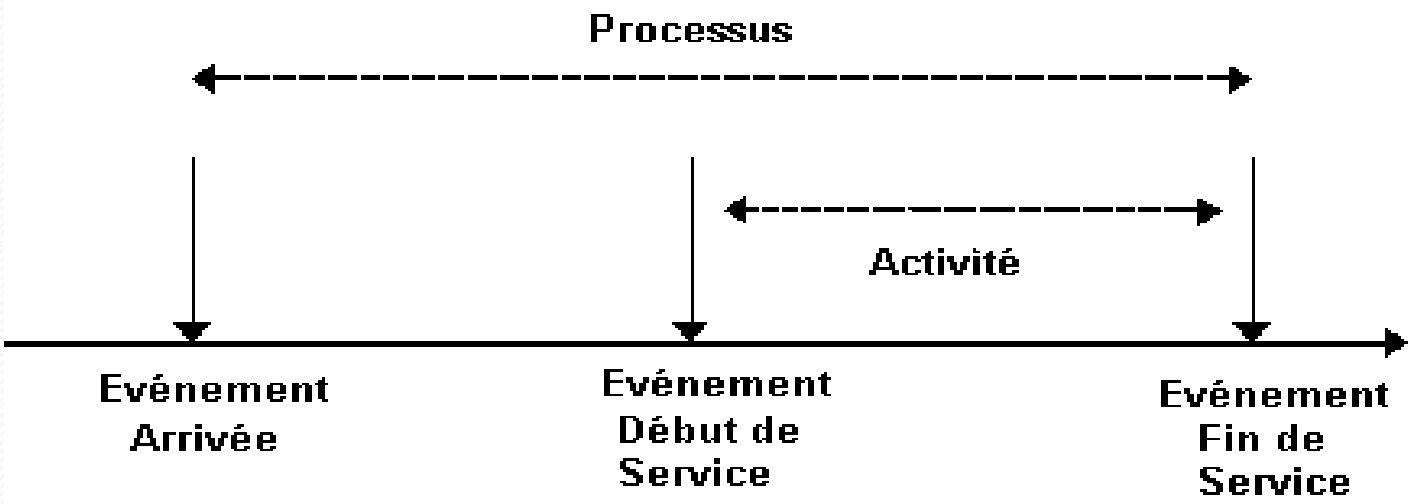
- **Processus**

Un processus est le regroupement d'une séquence d'événements dans l'ordre chronologique dans lequel ces événements se produiront. Comme les événements peuvent initier des activités, un processus peut inclure aussi des activités.

Exemple :

- L'arrivée d'un client , sa mise en file d'attente, son début de service , peuvent constituer un processus associé au client.

- **Relation entre Evénement, Activité, Processus**



Construction d'un modèle a événements discrets

- Pour construire un modèle de simulation à événements discrets, le concepteur doit choisir une approche de modélisation
- Si le concepteur a à sa disposition un langage de simulation, dans ce cas l'approche à utiliser est implicite puisque chaque langage offre normalement une (ou plusieurs approches).
- Si par contre, le concepteur utilise un langage de programmation général (Fortran, ...), alors le choix d'une approche est sous son entière responsabilité.

- Les trois approches les plus connues sont :

1) L'approche par evennement
2) L'approche par activité
3) L'approche par interaction de processus

Construction d'un modèle a événements discrets

Ces trois approches de modélisation sont caractérisées par le fait qu'elles conduisent à des modèles ayant une structure hiérarchique à 3 niveaux qui sont :

- **niveau 1 :** Correspond au programme de contrôle de la simulation appelé aussi exécutif ou noyau

• Le noyau est responsable du séquencement des opérations (événements, activités, processus) qui ont lieu au fur et à mesure que la simulation progresse. Ainsi, le noyau contrôle le niveau 2 et comprend des routines chargées de :

- Identifier quand aura lieu le prochain événement
- S'assurer que les bonnes opérations aient lieu à cet instant

• Dans le cas ou on dispose d'un langage de simulation, le noyau est une partie qui n'est pas directement accessible au programmeur qui n'a vraiment pas à savoir tous les détails sur le noyau.

Par contre avec un langage de programmation général(Fortran, Pascal, C,...), il faudra écrire soit même le noyau et il faudra dans ce cas maîtriser tous les détails de fonctionnement du noyau.

Construction d'un modèle a événements discrets

- **niveau 2 :** Ce niveau est à la charge du programmeur est constitue le modèle de simulation du point de vue du programmeur. C'est donc un ensemble d'instructions dont la structuration dépend de l'approche de modélisation adoptée. Il s'agira de routines décrivant des événements dans le cas d'une approche par événements, de routines décrivant des activités dans le cas d'une approche par activités ou de routines décrivant des processus dans le cas d'une approche par processus.

- **niveau 3 :** Comprend un ensemble de routines utilitaires appelées par le niveau 2. Il s'agit de routines pour générer des nombres aléatoires, collecter des statistiques, produire des résultats sous forme de rapport, etc... Le programmeur fait appel à ces routines au moment de l'écriture du niveau 2.

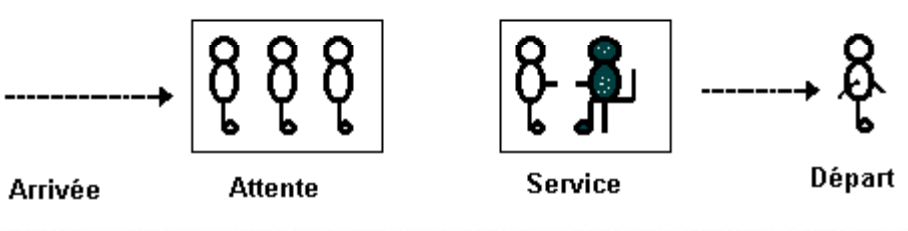
Approches de modélisation des SAED

➢ APPROCHE PAR EVENEMENTS

Dans cette approche, un système est modélisé en définissant les changements qui ont lieu aux instants d'occurrence des événements. Le concepteur doit donc déterminer les événement qui peuvent changer l'état du système et de développer ensuite la logique associée avec chaque type d'événement.

Exemple:

Considérons comme système à modéliser une banque dans laquelle arrivent des clients qui sont servis par un seul serveur (employé). Les clients arrivent à la banque, se mettent en attente devant un guichet, sont servis puis quittent la banque.



L'état du système est défini par l'**état du serveur** et par le **nombre de clients en attente**. Ainsi, l'état reste constant sauf lorsqu'un **client arrive** ou lorsqu'un **client quitte** la banque.

L'approche par événements consiste à décrire ce qui se passe lorsqu'un **client arrive** et lorsqu'un **client termine** le service. Du fait que le système change d'état uniquement

aux instants de ces deux événements (Arrivée d'un client ou départ d'un client) , ces derniers suffisent pour décrire complètement la dynamique de ce système.

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**

Approches de modélisation des SAED

Evénement : **Fin_De_Service**

Ejecter le client du système ;
Si **Clients_En_Attente** > 0
Alors

/* 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

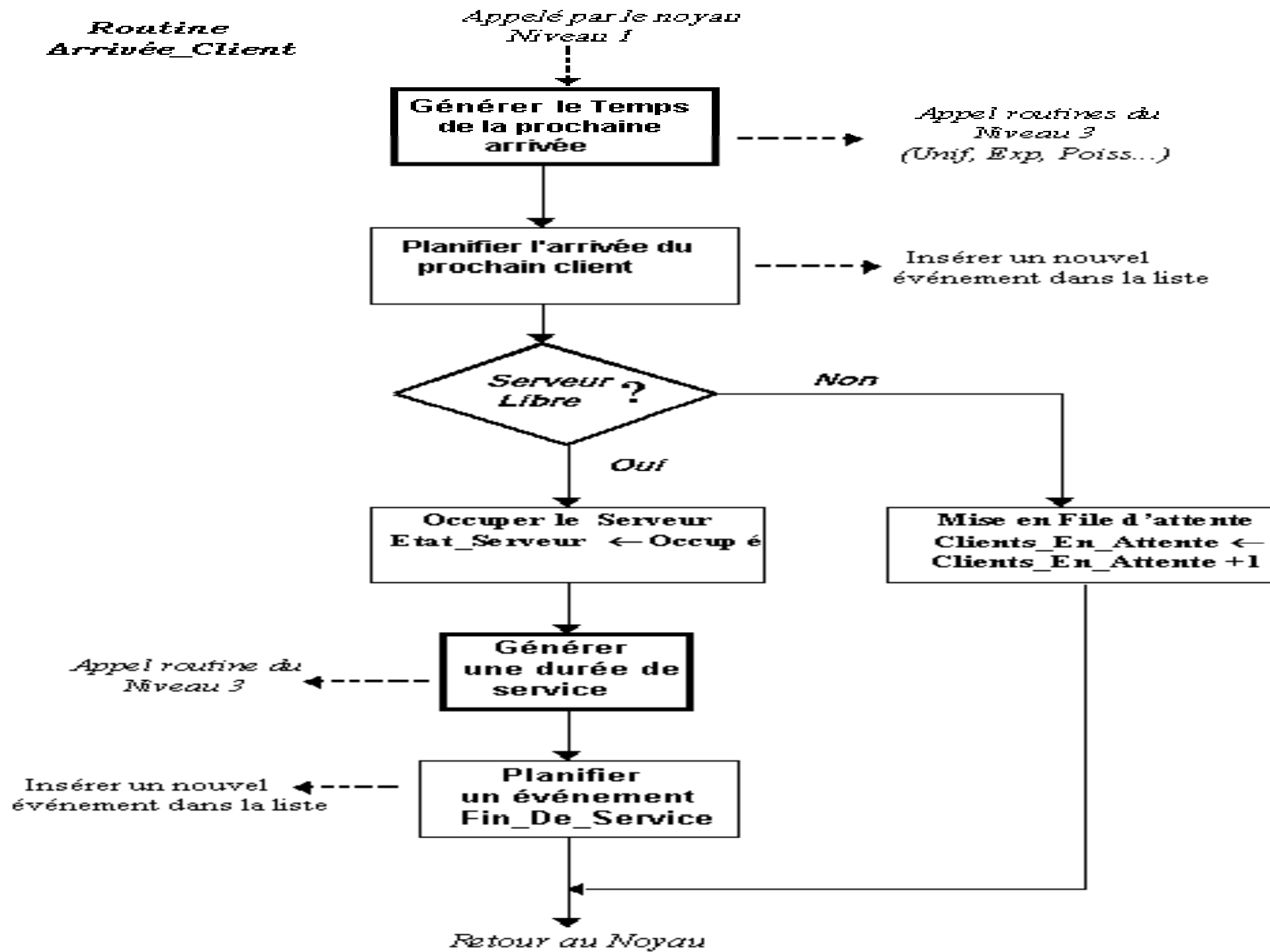
Sinon

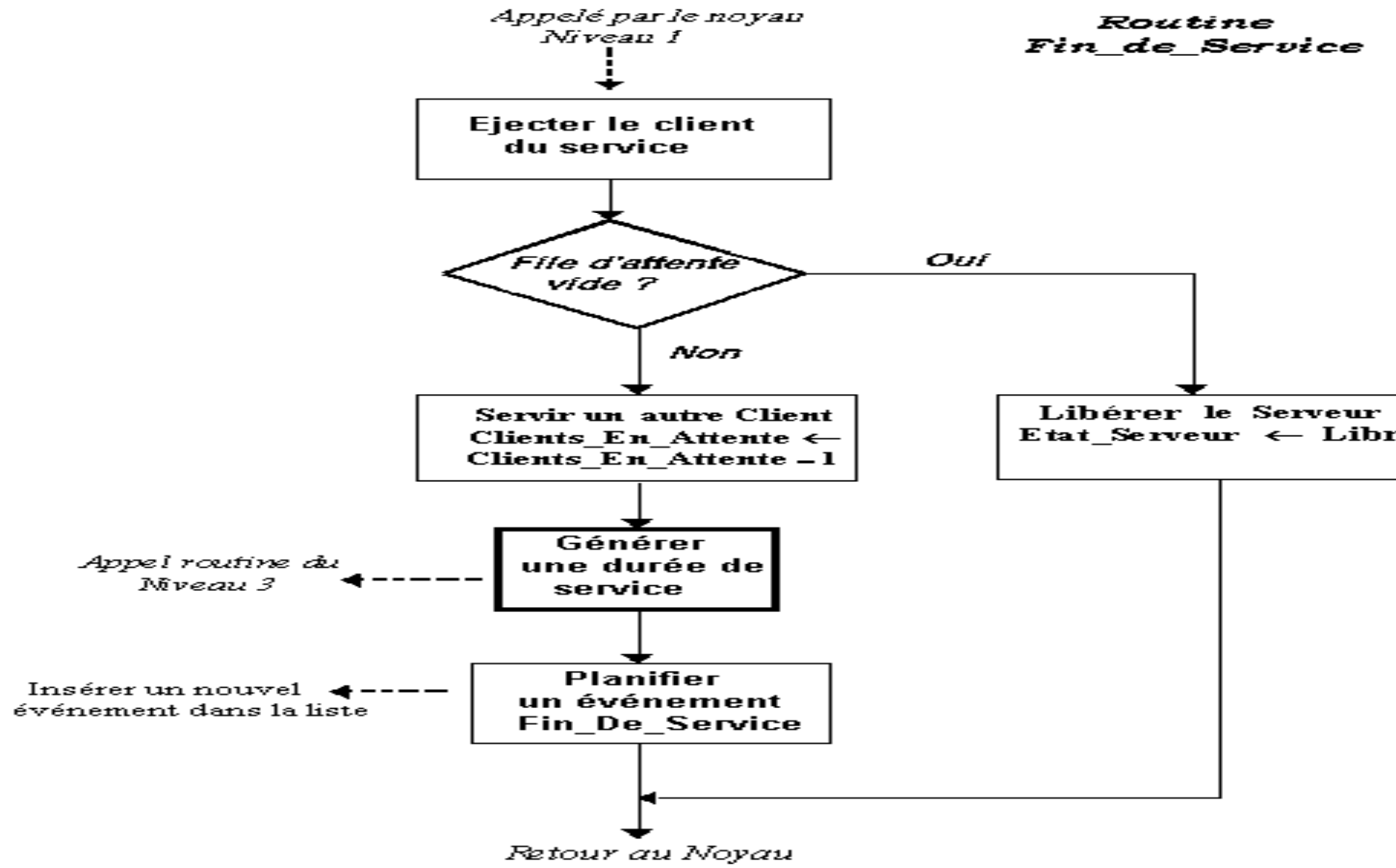
/* le serveur vient de finir de servir un client et */
/* *il n' y a plus de clients en attente* */

- **ETAT_SERVEUR** ← Libre

Fin

Fin événement **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
Appel routine du Niveau 3
Générer une durée de service
Insérer un nouvel événement dans la liste
Planifier un événement Fin_De_Service
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 ← Libr
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

Approches de modélisation des SAED

➢ Dans l'approche par événements, les tâches principales du noyau seront :

1) Contrôler le temps : déterminer la date du prochain événement et initialiser l'horloge avec cette date

2) Identifier les événements : déterminer quels sont les événements qui doivent avoir lieu au temps courant (horloge)

3) Exécuter les événements : déclencher les événements identifiés comme devant avoir lieu

Exemple :

- L'arrivée d'un client provoquera l'ajout d'un événement **Fin_De_Service** à l'agenda (appelé aussi calendrier ou échéancier). Elle provoque aussi l'ajout d'un événement **Arrivée_Client** correspondant au prochain client
- La fin d'un service peut provoquer l'ajout d'un événement **Fin_De_Service** à l'agenda dans le cas ou la file d'attente n'est pas vide et qui concerne dans ce cas le nouveau client pris dans la file.
- Ainsi, chaque événement de la liste doit comprendre au moins les deux informations suivantes :
  - la date d'occurrence de l'événement
  - l'identification de l'événement (en général un numéro)
- Des informations sur les entités impliquées par cet événement peuvent être aussi utiles (ex : choix d'un client dans la file).
- Au fur et à mesure que la simulation progresse, le noyau va exécuter un cycle à 2 phases:

Approches de modélisation des SAED

1) contrôle du temps : cette phase comprend

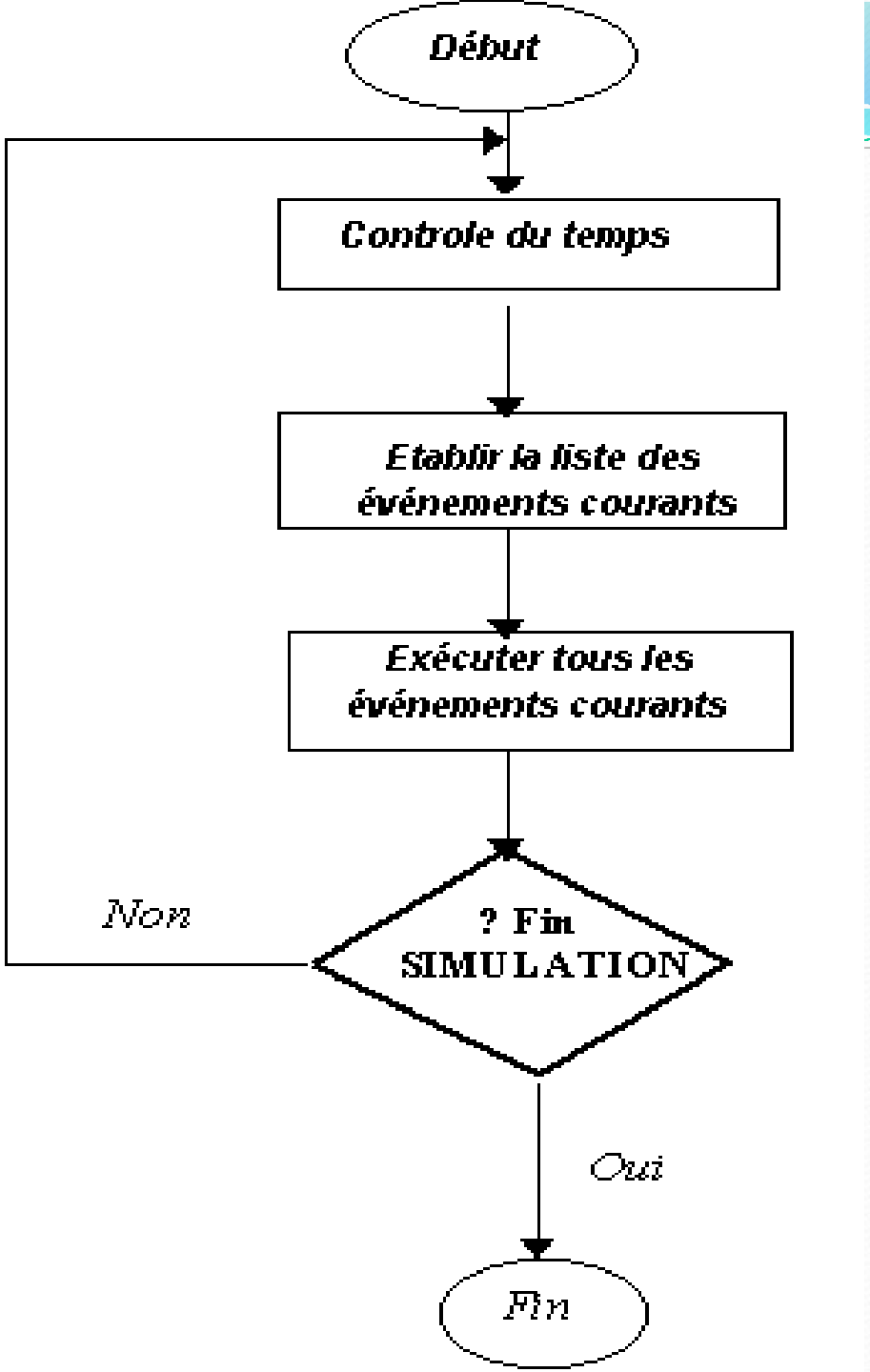
a) la détermination de la date du prochain événement en examinant le calendrier (liste des événements)

b) l'initialisation de l'horloge avec la date du prochain événement

c) la construction d'une liste des événements courants comprenant tous les événements dont la date d'occurrence est égale à l'horloge

2) Exécution des événements courants

- Les événements courants sont exécutés sous le contrôle du noyau. Aucun événement ne peut être déclenché sans le noyau. Ceci garantit que l'enchaînement logique des événements est entièrement contrôlé par le noyau. Une fois un événement exécuté, il est détruit (du calendrier et de la liste des événements courants)
- Ce cycle est répété jusqu'à la fin de la simulation. Un organigramme simplifié du noyau pourrait être :


Début
Controle du temps
Etablir la liste des événements courants
Exécuter tous les événements courants
? Fin SIMULATION
Non
Oui
Fin

Approches de modélisation des SAED

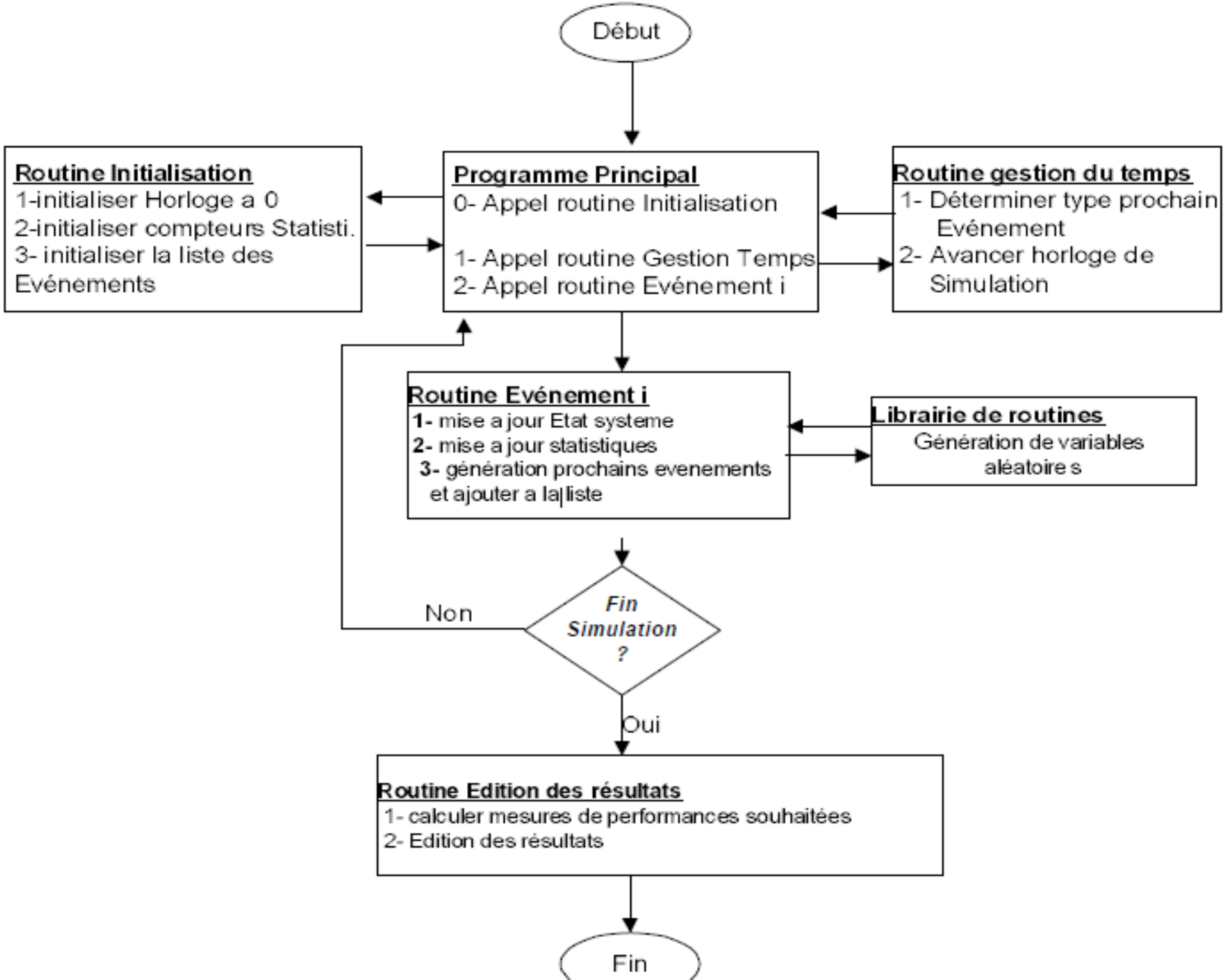
➢ **Composantes d'un modèle de simulation a événements discrets**

les composantes suivantes se retrouvent dans la plupart des modèles de simulation a événements discrets adoptant l'approche par ''événement'':

1. **L'état du système:** défini par une collection de variables d'état décrivant l'état du système à a un temps particulier.
2. **L'horloge de la simulation:** variable indiquant la valeur courante du temps de la simulation.
3. **La liste des événements:** Une liste qui contient les dates d'occurrence des événement devant avoir lieu dans le futur.
4. **Des compteurs statistiques:** Variables utilisées pour collecter des informations statistiques sur les performances du système.
5. **La routine d'initialisation.** Sous programme servant a initialiser le modèle de simulation au temps zéro.
6. **Routine de Gestion du temps (contrôle du temps):** sous programme qui détermine le prochain événement a partir de la liste des événements et initialise l'horloge de simulation avec le date d'occurrence de cet événement courant.

Approches de modélisation des SAED

7. **Routines associées aux événements:** sous programme qui met à jour l'état du système quand un type particulier d'événement se produit (il y a une routine d'événement pour chaque type d'événement).
8. **bibliothèque de routines utilitaires:** ensemble de sous programmes utilisés pour générer des variables aléatoires identifiées comme faisant partie du modèle de simulation.
9. **Générateur des résultats:** sous programme qui calcule des estimations (a partir des compteurs statistiques) des mesures de performance désirées et génère en fin de simulation un rapport.
10. **Le programme principal:** chargé d'intialiser l'état du systeme au début de la simulation, toute les variables utilisées au cours de la simulation. Il invoque la routine de gestion du temps pour déterminer le prochain événement devant avoir lieu et passe le contrôle à la routine associée a cet événement. Il contrôle aussi si la fin de la simulation est atteinte et invoque dans ce cas le générateur de resultats


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 aléatoires
Fin Simulation ?
Non
Oui
Routine Edition des résultats
1- calculer mesures de performances souhaitées
2- Edition des résultats
Fin

Les langages de simulation

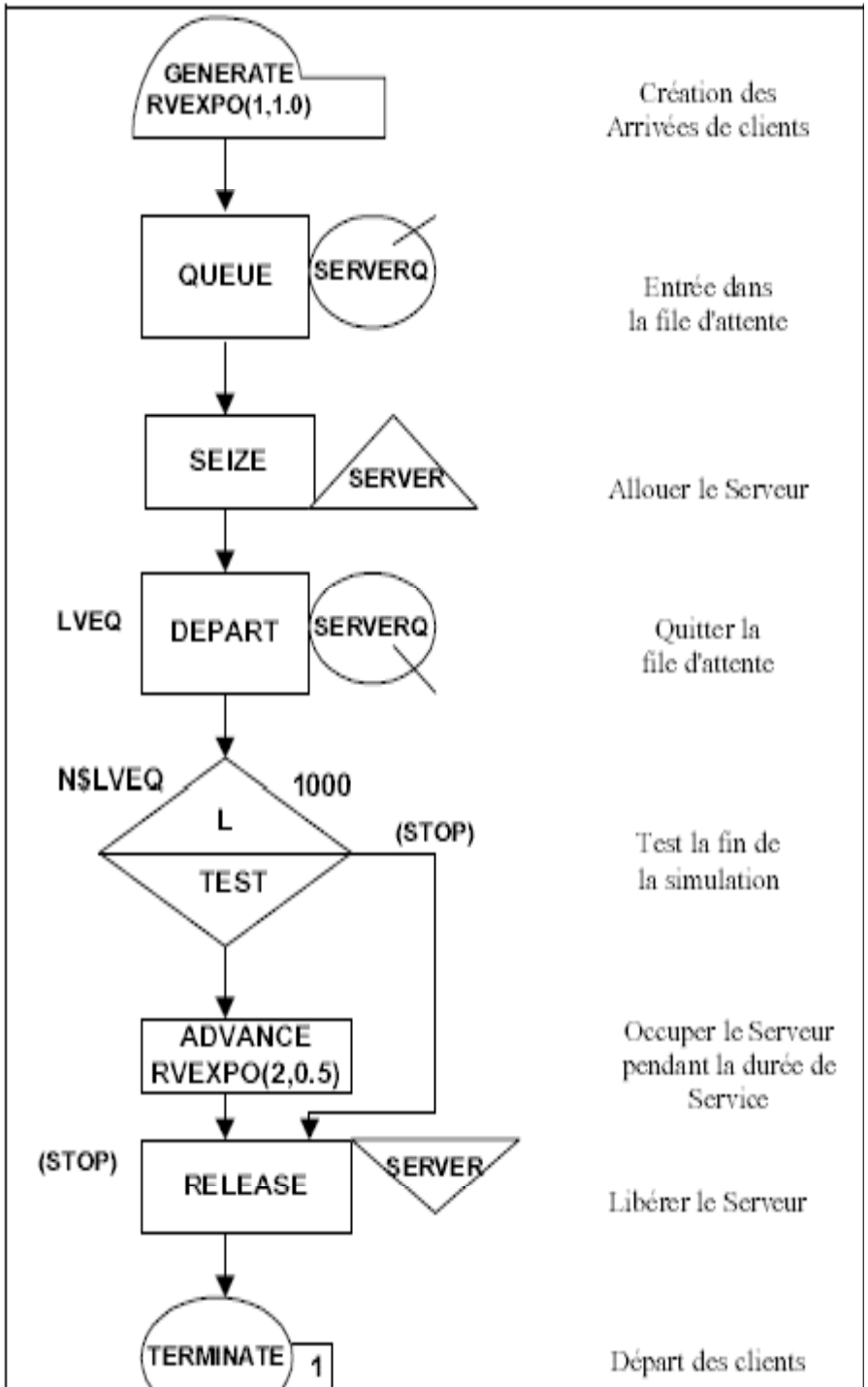
Les premiers langages de simulation ont vu le jour vers 1960, et étaient plus orientés vers la représentation et la simulation des systèmes discontinus (à événements discrets).

Ils permettent de programmer des modèles sous-tendus par des concepts indépendants des domaines d'application.

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

Les langages de simulation

➢**GPSS(General Purpose Simulation System)** est un langage offrant une approche orientée processus. Il a été développé initialement chez IBM en 1961, puis plusieurs versions ont vu le jour dont la plus récente disponible chez IBM est GPSS/V. GPSS est un des précurseur de l'approche de modélisation par "processus " et possède un support graphique de représentation comme la plupart des langages actuels.




```
1    *
2    *          Exemple de  Modèle en GPSS
3    *
4         SIMULATE
5         GENERATE       RVEXPO(1,1.0)      creation -Injection des entités dans
                                            système
6         QUEUE          QSERVER            entrée dans la file d'attente
7         SEIZE          SERVER             demande d'allocation du serveur
8    LVEQ DEPART         QSERVER            sortie de la la file d'attente
9         TEST L         N$LVEQ,1000,STOP   Test si l'execution est arrivée a sa fin
10        ADVANCE        RVEXPO(2,0.5)      occuper le serveur pendant t = durée
                                            service
11   STOP RELEASE        SERVER             libérer le serveur
12        TERMINATE      1                  entité quitte le système
13   *
14   *    Instructions   de Controle
15   *
16        START          1000               faire 1 seule simulation qui dure 1000 unités
17        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 --

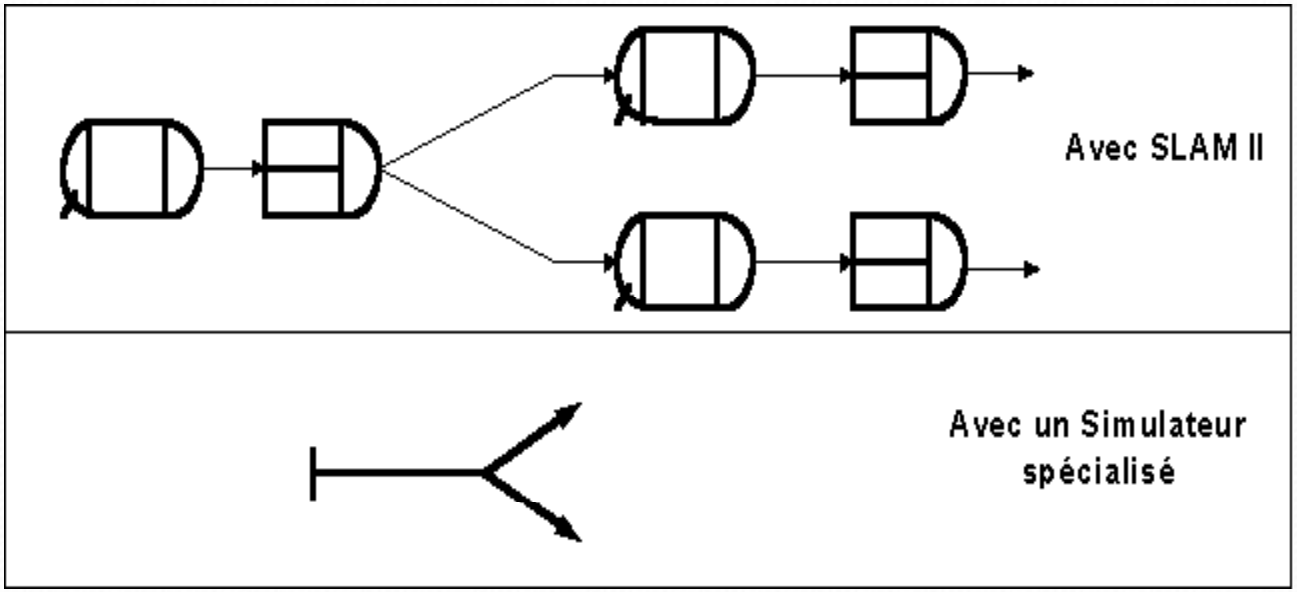
FACILITY	TOTAL TIME	AVAIL TIME	UNAVL TIME	ENTRIES	AVERAGE TIME/XACT	CURRENT STATUS	PERCENT AVAIL	SEIZING XACT	PREEMPTING XACT
SERVER	0.516			1000	0.523	AVAIL			

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/UNIT	SAVERAGE 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

Les simulateurs spécialisés

- Un simulateur spécialisé est un outil de simulation offrant un langage de description (ou un système d'entrées interactif de données) dans lequel les instructions (ou primitives) sont des objets du système a modéliser. Les primitives offertes sont, dans ce cas, des agrégats de processus élémentaires (files d'attente, ressources, activités, etc ...) représentant un objet particulier du système (machine, stock, convoyeur, palette, etc ... dans le cas de système de production) et son comportement.
- Un des avantages des simulateurs spécialisés est la réduction des efforts de conceptualisation à fournir dans l'étape de modélisation. La figure suivante permet de montrer la simplicité de représentation d'un même problème (ici la modélisation d'un aiguillage divergent du réseau de manutention) en utilisant un simulateur spécialisé "réseaux de manutention", et un langage de simulation général (ici SLAM II).



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

Les simulateurs spécialisés

- **NETWORK II.5** est un simulateur dédié pour les systèmes informatique en général et plus particulièrement adapté pour la modélisation des réseaux locaux de communication (Local Area Networks). Les blocs de bases offerts pour construire un modèle sont les unités de traitements (par exemple, un CPU), les unités de transfert de données (par exemple, un bus), les unités de stockage (exemple, une unité de disques), et les modules logiciel.

- **COMNET II.5,** est aussi un simulateur dédié pour les réseaux de télécommunication

larges(Wide Area Networks). Les blocs de bases offerts pour construire un modèle sont la topologie du réseau (noeuds avec leurs connexions), le traffic dans le réseau (source,

destination, et taille des messages), et les opérations dans le réseau (stratégies de routage des messages). Ces deux simulateurs sont distribués par la societé CACI (los Angeles).