



Chapitre 4

Simulation à événements discrets

2025-2026

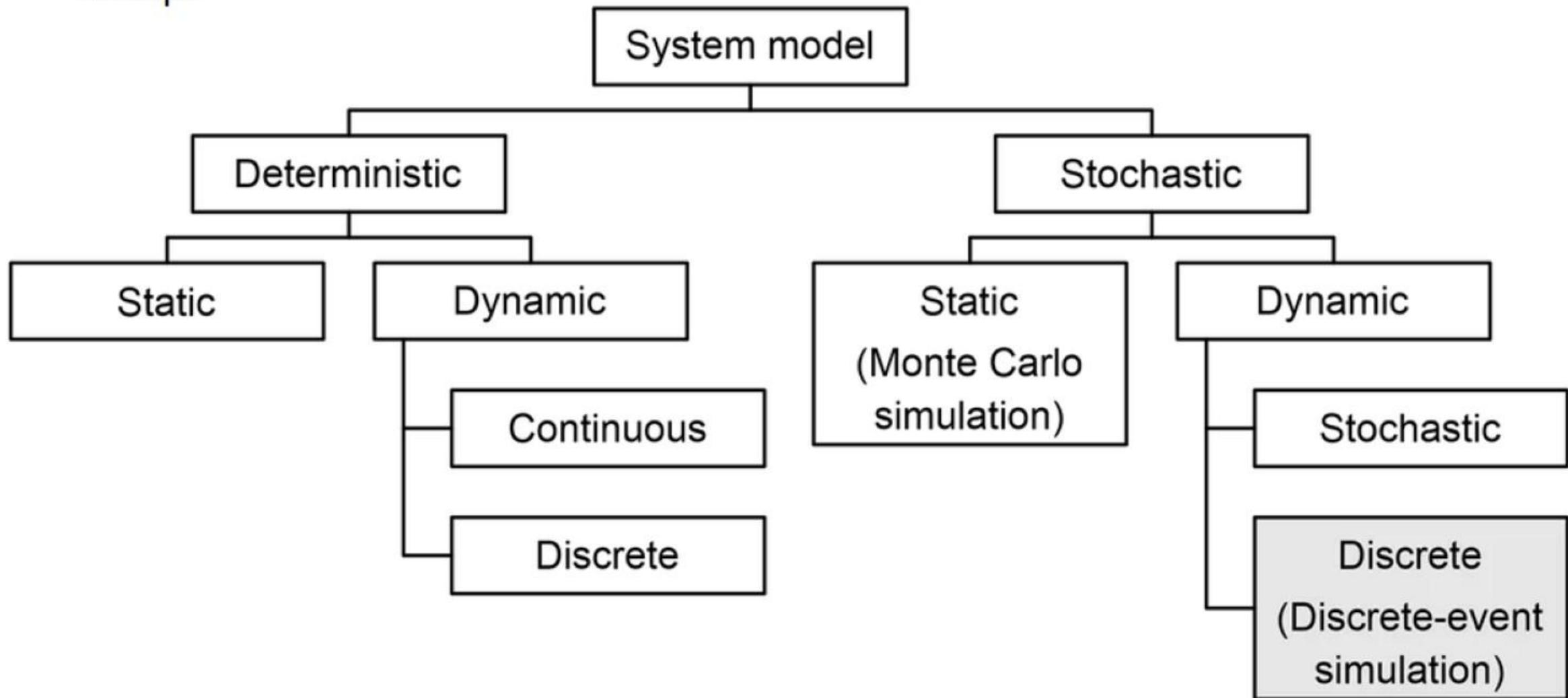
Niveau : L2-INFO

hajmessaoud.nada@gmail.com

Dr. Nada Haj Messaoud

Taxonomie des modèles de simulation

Recap:



Taxonomie des modèles de simulation

Recap:

1. Déterminer les objectifs → Étape 1 : Analyse du problème
 2. Élaborer le modèle conceptuel
 3. Établir le modèle de spécification
 4. Développer le modèle informatique
 5. Vérification du modèle
 6. Validation du modèle
 7. Analyse des résultats + conclusions → Étape 4 : Rapport et conclusions
-] Étape 2 : Modélisation et programmation
] Étape 3 : Expérimentations sur le modèle

Plan

- Concepts de la simulation à événements discrets
 - Terminologie et notions de base
 - Exemples
- Composants d'une simulation à événements discrets
 - Méthodes d'avancement du temps
 - Approche par calendrier d'événements (Event scheduling approach)
- Simulation manuelle
 - Exemple supermarché
- Programmation
 - Simulation des systèmes de files d'attente
 - Modèle à population infinie et modèle à population finie
 - File d'attente en tandem avec blocage
- Vérification et validation des modèles de simulation

Plan

- Concepts de la simulation à événements discrets
 - Terminologie et notions de base
 - Exemples
- Composants d'une simulation à événements discrets
 - Méthodes d'avancement du temps
 - Approche par calendrier d'événements (Event scheduling approach)
- Simulation manuelle
 - Exemple supermarché
- Programmation
 - Simulation des systèmes de files d'attente
 - Modèle à population infinie et modèle à population finie
 - File d'attente en tandem avec blocage
- Vérification et validation des modèles de simulation

Concepts de la simulation à événements discrets (Partie 1)

- **Modèle** : représentation abstraite d'un système (réel)
- **Système** : ensemble d'entités qui interagissent entre elles au cours du temps (ex. : personnes, machines, processeur, serveur web)
- **Etat Système**: ensemble des variables qui contiennent toute l'information nécessaire pour décrire complètement le système à un instant donné (ex. : nombre de clients dans la file, état occupé/libre du serveur)
- **Entité** : tout objet ou composant du système (ex. : un client, un serveur, une machine, un paquet réseau)
- **Attributs** : propriétés ou caractéristiques d'une entité donnée (ex. : heure d'arrivée du client, priorité, type de panne)
- **Liste** : collection d'entités associées, ordonnées selon une logique particulière (ex. : files d'attente, ensembles, listes d'événements)

Concepts de la simulation à événements discrets (Partie 2)

- **Évènement** : occurrence instantanée qui modifie l'état du système (ex. : arrivée d'un nouveau client, fin de service, panne d'une machine)
- **Liste des évènements futurs**: ou calendrier des événements (on dit le plus souvent : calendrier d'événements ou liste des événements futurs (LEF))
- **Activité** : durée de temps de longueur connue et fixée dès son commencement (ex. : temps de service d'un client, temps de traitement d'une pièce)
- **Délai** : durée de temps de longueur indéterminée, non connue tant qu'elle n'est pas terminée (ex. : temps d'attente d'un client dans la file)
- **Horloge** : variable représentant le temps simulé (peut avancer de façon continue ou par sauts discrets selon le type de simulation)

Concepts de la simulation à événements discrets (Partie 2)

Concepts clés

- **Activité** : représente un temps de service, un temps inter-arrivée ou tout temps de traitement dont la durée a été définie ou caractérisée au préalable par le modélisateur. La durée d'une activité peut être :
 - déterministe (fixe) ou stochastique (aléatoire)
 - une fonction dépendant des variables du système et/ou des attributs des entités
- **La durée d'une activité n'est pas influencée par la survenue d'autres événements** (elle est connue dès qu'elle commence).
- **Retard (ou délai)** : durée non connue à l'avance et déterminée par l'état actuel du système (le modélisateur ne la fixe pas au départ). Exemple : le temps d'attente d'un client dans une file dépend du nombre de clients devant lui, de leurs temps de service, d'une éventuelle panne du serveur et de son temps de réparation, etc.

Exemple : Centre d'appels ABC

Exemple 1: Centre d'appels ABC

Un centre de support technique téléphonique avec trois techniciens qui prennent les appels et assurent le service :

- Trois agents : **Alice, Bob, Chris** (canaux de support multiples)
- Règle de simplification en cas d'égalité : priorité alphabétique lorsqu'au moins deux agents sont libres (Alice avant Bob avant Chris)

Objectif de la simulation : Évaluer les performances de l'organisation actuelle, en particulier **le temps de réponse du système** (temps moyen d'attente + temps de service).

Variables aléatoires du modèle :

- Temps inter-arrivée entre deux appels (généralement exponentielle)
- Temps de service (lois différentes selon l'agent : Alice, Bob ou Chris peuvent avoir des durées moyennes ou des distributions différentes)

Exemple : Centre d'appels ABC

États du système

Le modèle du centre d'appels ABC est une simulation à événements discrets dont l'état **du système** à l'instant t est complètement décrit par les variables suivantes :

Variables d'état du système :

- Nombre d'appels (clients) en attente à l'instant t
- Indicateur d'état d'Alice : libre (idle) ou occupé (busy) à l'instant t
- Indicateur d'état de Bob : libre ou occupé à l'instant t
- Indicateur d'état de Chris : libre ou occupé à l'instant t

Entités : Ni les appellants ni les techniciens n'ont besoin d'être représentés explicitement (on peut se contenter des variables d'état ci-dessus), **sauf** si l'on souhaite collecter des statistiques individuelles par appelant (ex. : temps d'attente de chaque client) ou par technicien (ex. : taux d'occupation individuel d'Alice, Bob ou Chris).

Exemple : Centre d'appels ABC

Événements, activités et retards

- Événements (occurrences instantanées qui changent l'état du système) :
 - Arrivée d'un appel
 - Fin de service par Alice
 - Fin de service par Bob
 - Fin de service par Chris
- Activités (durées connues dès leur début) :
 - Temps inter-arrivée entre deux appels
 - Temps de service assuré par Alice
 - Temps de service assuré par Bob
 - Temps de service assuré par Chris
- Retard (ou délai) (durée inconnue à l'avance, dépend de l'état du système) :
 - Temps d'attente d'un appelant dans la file jusqu'à ce qu'Alice, Bob ou Chris se libère

Exemple 2 : Restaurant de pancakes

Exemple 2 : Restaurant de pancakes

Un restaurant très original installé dans une ancienne église propose le déroulement suivant :

- **Hôte/Hôtesse** : accueil et placement des clients à une table (possibilité d'attente à l'entrée si toutes les tables sont occupées)
- **Serveur/Serveuse** : prise de commande, service des plats et boissons
- **Cuisine et cuisinier(s)** : préparation des commandes (file d'attente possible en cuisine)
- **Caissier/Caissière** : encaissement et départ des clients

Objectif : Déterminer le nombre optimal de tables et de personnel (hôtes, serveurs, cuisiniers, caissiers) afin de maintenir un temps de réponse global raisonnable (temps total passé par un client dans le restaurant).

Variables aléatoires :

- Horaires d'arrivée des clients
- Taille des groupes
- Heure de la journée
- Temps de prise de commande
- Temps de préparation en cuisine
- Temps de repas
- Temps d'encaissement, etc.

Exemple 2 : Restaurant de pancakes

Exemple 2 : Restaurant de pancakes

Description du modèle à événements discrets :

- **État du système :**
 - Nombre de clients en attente d'être placés (file à l'entrée)
 - Nombre de clients assis attendant de passer commande
 - Nombre de clients dont la commande est en préparation (en attente de nourriture)
 - Nombre de clients en train de manger
 - Nombre de clients en attente de payer
 - Nombre de tables libres / occupées à l'instant t
- **Entités :** Clients, Hôte / hôtesse, Serveurs / serveuses, Cuisiniers, Tables du restaurant, Éventuellement : caissier / caisse, autres?

Exemple 2 : Restaurant de pancakes

Exemple 2 : Restaurant de pancakes

- **Événements :**
 - Arrivée d'un client ou d'un groupe de clients
 - Fin du placement par l'hôte/hôtesse
 - Fin de prise de commande / service par le serveur/la serveuse
 - Fin de préparation par le cuisinier
 - Fin d'encaissement par le caissier
- **Activités :**
 - Temps inter-arrivée entre deux clients/groupes
 - Temps de placement (hôte/hôtesse)
 - Temps de prise de commande et de service (serveur/serveuse)
 - Temps de préparation en cuisine (cuisinier)
 - Temps d'encaissement (caissier)
- **Retards :** Temps d'attente pour être : placé à une table / avant que le serveur prenne la commande /pour recevoir les plats / pour payer

Plan

- Concepts de la simulation à événements discrets
 - Terminologie et notions de base
 - Exemples
- Composants d'une simulation à événements discrets
 - Méthodes d'avancement du temps
 - Approche par calendrier d'événements (Event scheduling approach)
- Simulation manuelle
 - Exemple supermarché
- Programmation
 - Simulation des systèmes de files d'attente
 - Modèle à population infinie et modèle à population finie
 - File d'attente en tandem avec blocage
- Vérification et validation des modèles de simulation

Composants d'une simulation à événements discrets

- Dans une simulation à événements discrets :
 - La simulation est pilotée par les événements
 - Le temps simulé avance d'événement en événement (selon l'ordre chronologique des événements)
 - L'état du système ne change qu'au moment précis où un événement se produit
- Exigences indispensables :
 - Algorithme d'avancement du temps
 - Calendrier d'événements (gestion de la liste des événements futurs – Future Event List)
 - Mécanisme de traitement (ou de processing) des événements

Méthodes d'avancement du temps

- La gestion correcte du temps de simulation et le respect strict de l'ordre chronologique des événements sont essentiels pour garantir la précision et le réalisme du modèle.
- **Deux grandes approches :**
 1. **Approche par pas de temps fixe (Time-Stepping)**
 - L'horloge de simulation avance d'un pas constant prédéfini (ex. : 1 seconde, 1 minute, 0,1 h).
 - À chaque pas, on vérifie toutes les activités et conditions du système pour voir si quelque chose doit se déclencher.
 - Si la condition est vérifiée, l'activité ou l'événement correspondant est déclenché.
 2. **Approche par calendrier d'événements (Event-Scheduling)**
 - Une liste des événements futurs (LEF ou FEL) contient tous les événements planifiés, triés par date de survenue.
 - L'horloge de simulation saute directement à l'instant du prochain événement le plus proche (on ignore complètement les périodes où rien ne se passe).
 - À chaque événement traité :→ mise à jour de l'état du système → éventuelle planification de nouveaux événements futurs.

Approche par pas de temps fixe

- À chaque pas de temps t , on parcourt la liste de tous les événements en attente pour vérifier lesquels satisfont leurs conditions et doivent se produire.
- La liste des événements futurs (LEF/FEL) n'est pas obligatoire et n'a pas besoin d'être triée, ce qui simplifie la programmation mais peut augmenter fortement le coût de calcul.
- **Taille du pas de temps:**
 - Si le pas est trop petit : surcharge de calcul
 - Si le pas est trop grand : plusieurs événements peuvent être déclenchés en même temps
- Dans la réalité, les intervalles entre événements varient énormément.
- L'approche par pas de temps fixe est très simple à comprendre, mais souvent très lente en exécution.
- **Systèmes petits avec peu d'événements prévisibles:**
 - Modèles de remboursement (paiements mensuels réguliers)
 - Simulations de flux de fluides à évolution très régulière

Approche par calendrier d'événements

- À tout instant t , la liste des événements futurs (LEF) ou calendrier d'événements (FEL) contient tous les événements déjà planifiés ainsi que leurs dates de survenue prévues (t_1, t_2, \dots, t_n).
- Cette liste est triée par ordre chronologique croissant des temps d'événement, et l'on a toujours : $t \leq t_1 \leq t_2 \leq \dots \leq t_n$ où t est la valeur actuelle de l'horloge de simulation (Clock).

Approche par calendrier d'événements

Etapes :

1. Retirer de la LEF la notice de l'événement imminent (ici l'événement de type 3 prévu à t_1)

2. Avancer l'horloge jusqu'à cet événement :
 $CLOCK \leftarrow t_1$

3. Exécuter l'événement imminent : mettre à jour l'état du système, modifier les attributs des entités, changer les appartennances si nécessaire.

4. Générer, si besoin, de nouveaux événements futurs et les insérer dans la LEF en respectant l'ordre chronologique (exemple : événement de type 4 prévu à t^* tel que $t_2 < t^* < t_3$).

5. Mettre à jour les statistiques cumulées et les compteurs

Ancien état du système à l'instant t

Horloge (CLOCK)	État du système	...	Liste des événements futurs (LEF)	...
t	$(5, 1, 6)$		$(3, t_1) \rightarrow$ Événement de type 3 à t_1 $(1, t_2) \rightarrow$ Événement de type 1 à t_2 $(1, t_3) \rightarrow$ Événement de type 1 à t_3 ... $(2, t_n) \rightarrow$ Événement de type 2 à t_n	

Nouvel état du système à l'instant t_1

Horloge (CLOCK)	État du système	...	Liste des événements futurs (LEF)	...
t_1	$(5, 1, 5)$		$(1, t_2) \rightarrow$ Type 1 à t_2 $(4, t^*) \rightarrow$ Type 4 à t^* $(1, t_3) \rightarrow$ Type 1 à t_3 ... $(2, t_n) \rightarrow$ Type 2 à t_n	

Traitement des listes

- **La gestion d'une liste**
 - Les principales opérations de traitement de liste effectuées sur une FEL (Future Events List) sont :
 - Suppression de l'événement imminent
 - Ajout d'un nouvel événement à la liste
 - Éventuellement suppression d'un événement (annulation d'un événement)
- **Structure de données pour la FEL:**
 - Variable(s)
 - Tableau(x)
 - Fichier(s)
 - Liste chaînée ordonnée
 - File de priorité (priority queue)
 - Tas binaire (binary heap)
 - File calendaire (calendar queue)

Liste des événements futurs

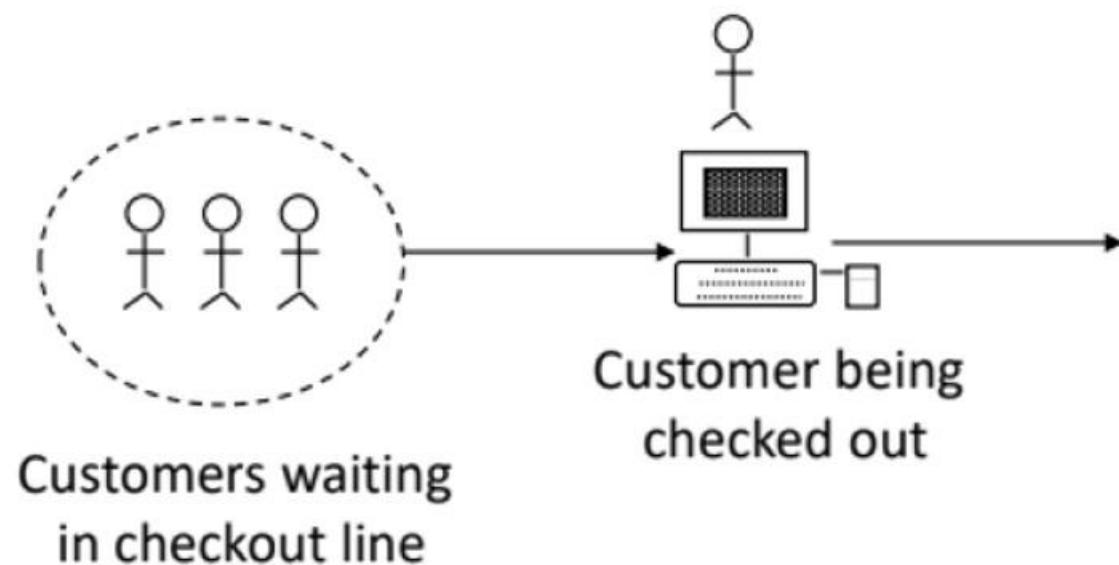
- **Événement d'arrivée:**
 - Par exemple, à l'instant 0, le premier événement d'arrivée est généré et planifié dans la FEL. Lorsque l'horloge de simulation avance finalement jusqu'à l'instant de ce premier arrivée, un deuxième événement d'arrivée est généré.
- **Événement de fin de service:**
 - Déclenché uniquement sous la condition qu'un client soit présent et qu'un serveur soit disponible.
- **Événement d'arrêt, E:**
 - À l'instant 0 : planifier un événement d'arrêt de la simulation à un instant futur spécifié TE.
 - La durée d'exécution TE est déterminée par la simulation elle-même. En général, TE correspond à l'instant de survenance d'un événement spécifié E (par exemple, le service du 1000e client) ou à la réalisation d'une condition particulière (par exemple, variation relative de l'estimation π).
22

Outline

- Concepts in discrete-event simulation
 - Terminology and concepts
 - Two pedagogical examples
- Components of discrete-event simulation
 - Time advance approaches
 - Event scheduling approach
- Manual simulation
 - Grocery store example
- Simulation program
 - Simulation of queuing systems
 - Infinite and finite population model
 - Tandem queue with blocking
- Verification and validation of simulation models

Exemple : Supermarché avec une seule caisse

- **File d'attente à canal unique (Single-channel queue):**
 - Le système comprend les clients en attente ainsi que le client éventuellement en cours de passage en caisse.
 - Pour cet exemple, une durée d'arrêt de simulation de 60 minutes est fixée.



Exemple : Supermarché avec une seule caisse

- Composants de l'exemple du supermarché

- État du système :
 - $LQ(t)$: nombre de clients en attente dans la file à l'instant t (excluant le client en cours de service à la caisse)
 - $LS(t)$: Nombre de clients en cours de service (1 ou 0) à l'instant t
- Entités: Le serveur et les clients ne sont pas modélisés explicitement, sauf en termes de variables d'état.
- Événements: Arrivée (A), départ (D), événement d'arrêt (E)
- Notices d'événements (type d'événement, heure de l'événement):
 - (A, t) représentant un événement d'arrivée prévu à l'instant futur t
 - (D, t) représentant le départ d'un client à l'instant t
 - (E, 60) représentant l'événement d'arrêt de la simulation à l'instant 60
- Activités: Temps inter-arrivée et temps de service
- Délai: Temps passé par un client dans la file d'attente

Exemple : Supermarché avec une seule caisse

- Scénario 1 dans l'exemple du supermarché

- Les conditions initiales sont les suivantes : le premier client arrive à l'instant 0 et commence immédiatement son service.
- Seules deux statistiques sont collectées ::
 - B: temps total d'occupation du serveur = B/TE
 - Max : longueur maximale observée de la file d'attente.
- Paramètres d'entrée
- Paramètres d'entrée :

Interarrival Times	0	8	6	1	8
Service Times	4	1	4	3	5

Exemple : Supermarché avec une seule caisse

- Scénario 1 dans l'exemple du supermarché

Interarrival Times	0	8	6	1	8
Service Times	4	1	4	3	5

Horloge	État du système		Liste des événements futurs	Commentaire	Statistiques cumulées	
	LQ(t)	LS(t)			B	MQ
0	0	1	(D, 4), (A, 8), (E, 60)	Première A se produit : ($a^* = 8$), planifier prochaine A : ($s^* = 4$) Planifier premier D	0	0
4	0	0	(A, 8), (E, 60)	Premier D se produit : (D, 4)	4	0
8	0	1	(D, 9), (A, 14), (E, 60)	Deuxième A se produit : (A, 8), ($a^* = 6$) Planifier prochaine A ; ($s^* = 1$) Planifier prochain D	4	0
9	0	0	(A, 14), (E, 60)	Deuxième D se produit : (D, 9)	5	0
14	0	1	(A, 15), (D, 18), (E, 60)	Troisième A se produit : (A, 14) ; ($s^* = 4$) Planifier prochain D	5	0
15	1	1	(D, 18), (A, 23), (E, 60)	Quatrième A se produit : (A, 15) (Client mis en attente)	6	1
18	0	1	(D, 21), (A, 23), (E, 60)	Troisième D se produit : (D, 18) ; ($s^* = 3$) planifier prochain D	9	1

Exemple : Supermarché avec une seule caisse

- Scénario 2 dans l'exemple du supermarché

Supposons que l'analyste de simulation souhaite estimer :

- le temps moyen passé dans le système (mean response time), et
- la proportion moyenne de clients qui passent 4 minutes ou plus dans le système (c'est-à-dire temps d'attente dans la file + temps de passage à la caisse).

Il est nécessaire d'étendre le modèle précédent pour représenter explicitement les clients individuels :

- Une entité « Client » dotée d'un attribut « heure d'arrivée » sera ajoutée à la liste des composants.
- Les entités clientes seront stockées dans une liste appelée « File d'attente à la caisse » (Checkout Queue) sous la forme C1, C2, C3, ...

Exemple : Supermarché avec une seule caisse

- Scénario 2 dans l'exemple du supermarché
 - Statistiques pour l'exemple du supermarché

Trois nouvelles statistiques cumulées seront collectées :

- **S** : la somme des temps de réponse des clients (temps total passé dans le système) pour tous les clients qui sont partis jusqu'à l'instant courant (c'est-à-dire \sum (heure de départ – heure d'arrivée) pour chaque client complété).
- **F** : le nombre total de clients qui ont passé 4 minutes ou plus à la caisse (temps d'attente dans la file + temps de service \geq 4 minutes).
- **ND** : le nombre total de départs (clients qui ont terminé leur passage à la caisse et quitté le système) jusqu'à l'instant courant de la simulation.

Exemple : Supermarché avec une seule caisse

- Tableau récapitulatif pour l'exemple du supermarché

■ Input parameters:

Interarrival Times	0	8	6	1	8
Service Times	4	1	4	3	5

Horloge	État du système		File d'attente à la caisse	Liste des événements futurs	Statistiques		
	LQ(t)	LS(t)			S	N_D	F
0	0	1	(C1, 0)	(D, 4, C1), (A, 8, C2), (E, 60)	0	0	0
4	0	0		(A, 8, C2), (E, 60)	4	1	1
8	0	1	(C2, 8)	(D, 9, C2), (A, 14, C3), (E, 60)	4	1	1
9	0	0		(A, 14, C3), (E, 60)	5	2	1
14	0	1	(C3, 14)	(A, 15, C4), (D, 18, C3), (E, 60)	5	2	1
15	1	1	(C3, 14), (C4, 15)	(D, 18, C3), (A, 23, C5), (E, 60)	5	2	1
18	0	1	(C4, 15)	(D, 21, C4), (A, 23, C5), (E, 60)	9	3	2

Exercice

Vous êtes responsable de la gestion d'un petit parking situé à proximité d'un bureau de poste (ou d'une banque). Ce parking dispose d'un maximum de **3 places de stationnement**. Le nombre de places est limité, et les clients (voitures) arrivent à des moments aléatoires pour trouver une place. Une fois qu'un client trouve une place, il y stationne pendant un certain temps avant de repartir.

Composants du système :

- **État du système :**

- **LQ(t)** : Nombre de voitures en attente à t (si le parking est plein).
- **LS(t)** : Nombre de places occupées à t ($0 \leq LS(t) \leq 3$).

- **Entités :**

- **Les voitures** sont les clients qui arrivent au parking et stationnent pendant un certain temps.
- **Le parking** est un espace limité avec seulement 3 places disponibles.

- **Événements :**

- **A(t)** : Arrivée d'un client au temps t. Si une place est disponible, le client occupe une place. Sinon, il attend.
- **D(t)** : Départ d'un client au temps t. Une voiture quitte le parking, libérant une place pour un autre client.
- **E (480)**: Fin de la simulation au temps t=480 (8 heures de simulation).

Temps inter-arrivées	0	5	2	7	4	6	3
Temps de stationnement	6	8	10	4	9	5	7

Exercice

- **Notifications d'événements :**

- **(A,t,C_i)**: Le client C_i arrive au temps t.
 - **(D,t,C_i)** : Le client C_i quitte le parking au temps t.
 - **(E,480)** : La simulation s'arrête au temps t=480.
-

Statistiques à collecter :

1. **S** : Somme des temps de réponse des clients ayant quitté le parking jusqu'à l'heure actuelle.
2. **F** : Nombre de voitures ayant passé **5 minutes ou plus** dans le parking.
3. **N_D** : Nombre total de départs avant la fin de la simulation.

Travail demandé : Dessiner le tableau de simulation pour obtenir les résultats demandés.

Exercice

Temps inter-arrivées	0	5	2	7	4	6	3
Temps de stationnement	6	8	10	4	9	5	7

Exercice

Temps inter-arrivées	0	5	2	7	4	6	3
Temps de stationnement	6	8	10	4	9	5	7

t	Event	Client	LQ(t)	LS(t)	Future Event List	S	F	ND
0	A	C1	0	1	(D,6,C1), (A,5,C2)	0	0	0
5	A	C2	0	2	(D,6,C1), (D,13,C2), (A,7,C3)	0	0	0
6	D	C1	0	1	(D,13,C2), (A,7,C3)	6	1	1
7	A	C3	0	2	(D,13,C2), (D,17,C3), (A,14,C4)	6	1	1
13	D	C2	0	1	(D,17,C3), (A,14,C4)	14	2	2
14	A	C4	0	2	(D,17,C3), (D,18,C4), (A,18,C5)	14	2	2
17	D	C3	0	1	(D,18,C4), (A,18,C5)	24	3	3
18	D	C4	0	0	(A,18,C5)	28	3	4
18	A	C5	0	1	(D,27,C5), (A,24,C6)	28	3	4
24	A	C6	0	2	(D,27,C5), (D,29,C6), (A,27,C7)	28	3	4
27	D	C5	0	1	(D,29,C6), (A,27,C7)	37	4	5
27	A	C7	0	2	(D,29,C6), (D,34,C7)	37	4	5
29	D	C6	0	1	(D,34,C7)	42	5	6
34	D	C7	0	0	(E,480)	49	6	7
480	E	-	0	0	-	49	6	7

Plan

- Concepts in discrete-event simulation
 - Terminology and concepts
 - Two pedagogical examples
- Components of discrete-event simulation
 - Time advance approaches
 - Event scheduling approach
- Manual simulation
 - Grocery store example
- Simulation program
 - Simulation of queuing systems
 - Infinite and finite population model
 - Tandem queue
- Verification and validation of simulation models

Programme de simulation générique

- **Initialisation**
 - Mettre l'horloge à zéro
 - Initialiser les variables d'état et les compteurs statistiques
 - Initialiser la liste des événements (avec les événements futurs déjà connus)
- **Boucle principale (répéter jusqu'à ce que la condition d'arrêt de la simulation soit satisfaite)**
 - Déterminer l'événement le plus imminent et le retirer de la liste des événements
 - Avancer l'horloge jusqu'à l'heure de cet événement
 - Appeler la routine d'événement correspondant au type i

Programme de simulation générique

- Routine d'événement (une routine séparée pour chaque type d'événement)
 - Mettre à jour les variables d'état
 - Mettre à jour les compteurs statistiques
 - Lorsque nécessaire, ajouter de nouveaux événements futurs à la liste des événements
- Générateur de rapport
 - Appelé lorsque la simulation est terminée
 - Calculer et afficher les mesures de performance qui intéressent l'analyste

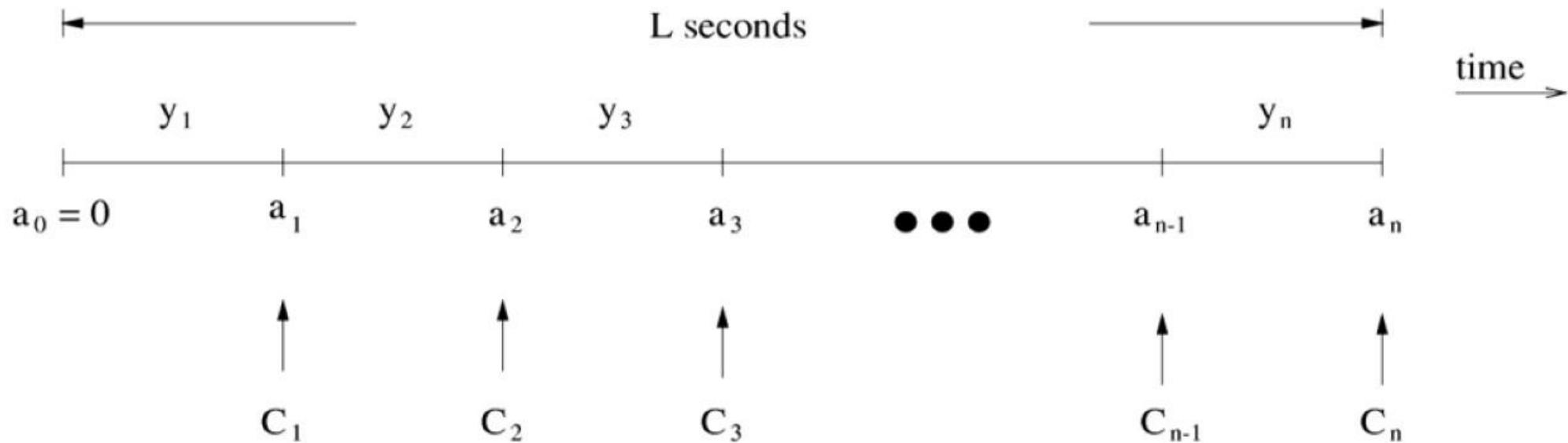
Simulation des systèmes de files d'attente

1. Serveur unique, population infinie
2. Serveur unique, population finie
3. Tandem queue

Arrival Definition

- Heure d'arrivée
 - Instant précis auquel un client arrive dans l'établissement de service
- Temps inter-arrivée
 - Durée écoulée entre deux arrivées successives de clients.
- Taux d'arrivée
 - Nombre moyen d'arrivées par unité de temps

Timing Diagram



C_j - customer j

a_j - arrival time of C_j

y_j - interarrival time between C_{j-1} and C_j

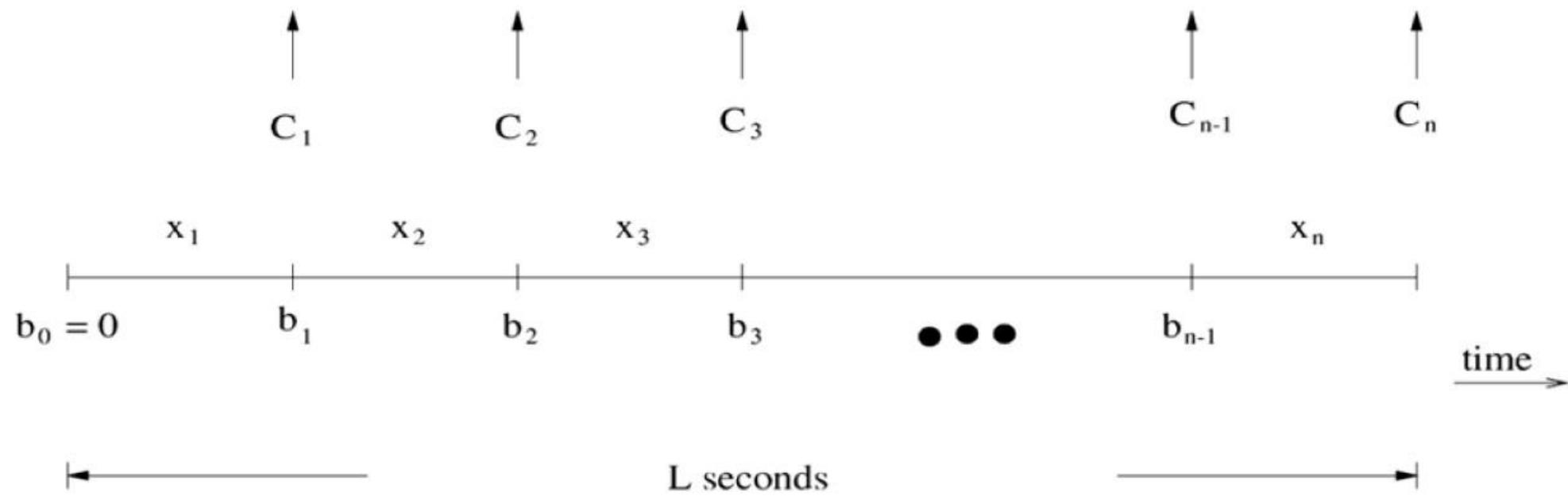
Relation entre le taux d'arrivée et le temps inter-arrivée

- Définition:

$$L = \sum_{j=1}^n y_j$$

- Temps inter-arrivée moyen $= \frac{L}{n}$ Avec n nombre d'arrivée - 1
- Taux arrivée $\frac{n}{L} = \frac{1}{\text{Temps inter-arrivée moyen}}$

Timing Diagram



C_j - customer j

x_j - service time of C_j

b_{j-1} - time at which C_j starts service

Relation entre le taux de service et le temps de service

- Définition :

$$L = \sum_{j=1}^n x_j$$

- Temps service moyen $= \frac{L}{n}$

- $Taux\ de\ service = \frac{n}{L} \frac{1}{mean\ service\ time}$

Mesures de performance typiques

- Temps de réponse
- Temps d'attente
- Nombre de clients dans le système
- Nombre de clients dans la file
- Utilisation du serveur
- Débit (Throughput)

Utilisation

- Proportion du temps pendant lequel le serveur est occupé



- Temps total d'occupation = $\sum_{j=1}^n s_j$
- U = Proportion du temps où le serveur est occupé = $\frac{1}{L} \sum_{j=1}^n s_j$

Note: $U \leq 1$

Throughput

- Taux auquel les clients quittent l'établissement de service après avoir terminé leur service
 - Débit:

$$R = \frac{n}{L}$$

où **n** est le nombre de clients servis pendant la période **L**

Exemple révisé : Calculer L, Temps inter-arrivée moyen, Taux d'arrivée, Temps total d'occupation, et Proportion du temps où le serveur est occupé

Données fournies :

- Heures d'arrivée des 5 tâches : 0, 2, 5, 8, 12
- Temps de service (s_i) pour les 5 tâches : 1, 2, 1,5, 2,5, 1

Étape 1 : Calculer L (Période totale d'observation)

Temps inter-arrivées = $2 - 0, 5 - 2, 8 - 5, 12 - 8 = 2, 3, 3, 4$

$L = 2 + 3 + 3 + 4 = 12$ unités de temps

Étape 2 : Calculer le Temps inter-arrivée moyen

En utilisant les temps inter-arrivées :

Temps inter-arrivée moyen = Somme des temps inter-arrivées / (Nombre de tâches - 1) = $12 / 4 = 3$
unités de temps

Exemple révisé : Calculer L, Temps inter-arrivée moyen, Taux d'arrivée, Temps total d'occupation, et Proportion du temps où le serveur est occupé

Étape 3 : Calculer le Taux d'arrivée (λ)

Le taux d'arrivée est l'inverse du temps inter-arrivée moyen :

$$\lambda = 1 / \text{Temps inter-arrivée moyen} = 1 / 3 \approx 0,33 \text{ tâches par unité de temps}$$

Étape 4 : Calculer le Temps total d'occupation (Total Busy Time)

Le temps total d'occupation est la somme de tous les temps de service :

$$\text{Temps total d'occupation} = s_1 + s_2 + s_3 + s_4 + s_5 = 1 + 2 + 1,5 + 2,5 + 1 = 8 \text{ unités de temps}$$

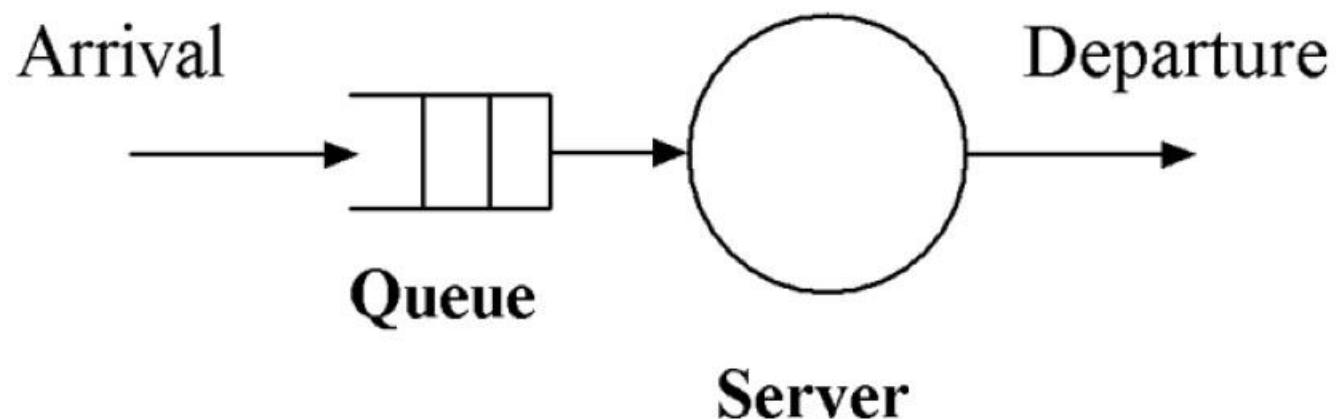
Étape 5 : Calculer la Proportion du temps où le serveur est occupé (U)

La proportion du temps où le serveur est occupé est :

$$U = \text{Temps total d'occupation} / L = 8 / 12 = 0,6667 (66,67 \%)$$

Exemple de file d'attente à serveur unique

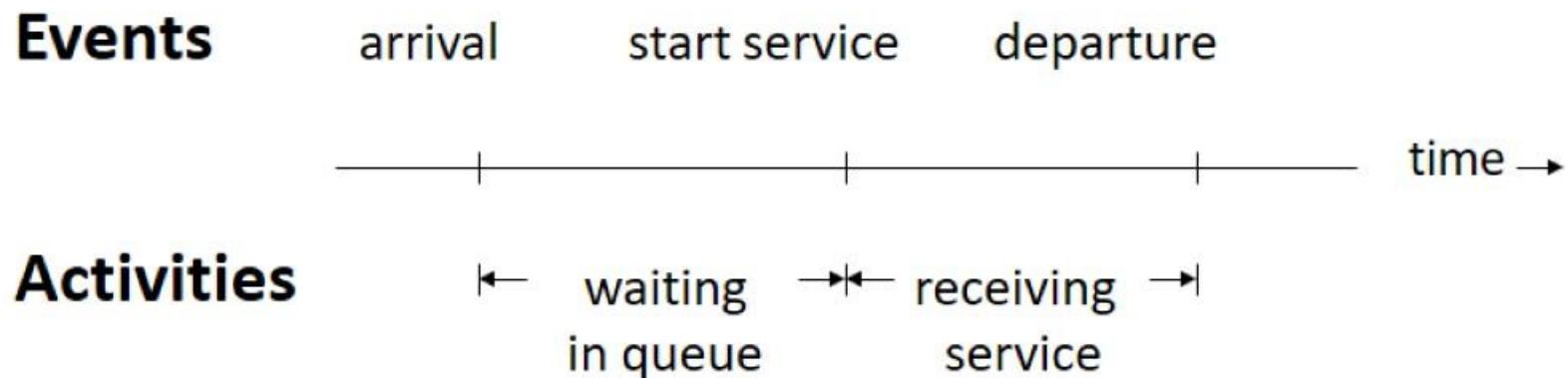
- Modèle à population infinie



Arrivées des clients - Modèle à population infinie

- Le nombre d'utilisateurs de l'établissement de service est important
- Le schéma des arrivées des clients est basé sur le comportement combiné des clients, et est supposé indépendant de l'état du système

Exemple de file d'attente à serveur unique



Modèle de file d'attente à serveur unique

■ Hypothèses

- Les temps inter-arrivées sont indépendants de l'état du système
- Les temps inter-arrivées sont iid (indépendants et identiquement distribués)
- Les temps de service sont indépendants de l'état du système
- Les temps de service sont iid (indépendants et identiquement distribués)
- Ordonnancement (premier arrivé, premier servi)
- Le système est vide au temps zéro
- L'arrivée du premier client se produit après le premier temps inter-arrivée
- La simulation se termine lorsque le m -ième client commence son service

Modèle de file d'attente à serveur unique

- Paramètres d'entrée :
 - Distribution des temps inter-arrivées (par ex., exponentielle)
 - Distribution des temps de service (par ex., uniforme)
- Mesures de performance d'intérêt
 - Temps d'attente moyen dans la file, \bar{W}
 - Nombre moyen de clients dans le système, \bar{n}

Modèle de file d'attente à serveur unique

■ Variables d'état

- status = état du serveur (busy ou idle)
- n = nombre de clients dans le système

■ Compteurs statistiques

- nW = nombre de temps d'attente accumulés
- sW = somme des temps d'attente accumulés
- sA = somme des aires accumulées (pour le calcul de \bar{n})
- last_event = instant du dernier événement lors du calcul de l'aire

Modèle de file d'attente à serveur unique

■ Listes

- event_list = liste des événements
- queue = file d'attente

■ Types d'événements

- type 1 : arrivée
- type 2 : début de service
- type 3 : départ

Modèle de file d'attente à serveur unique (1 / 4)

■ Initialisation

- clock = 0
- status = idle
- n = 0
- nW = sW = 0
- last_event = 0
- sA = 0
- Initialiser la file d'attente à vide
- Initialiser la liste des événements à vide
- Déterminer inter_t, le premier temps inter-arrivée
- Planifier un événement d'arrivée à clock + inter_t

Modèle de file d'attente à serveur unique (2 / 4)

■ Boucle principale (répéter jusqu'à ce que la condition d'arrêt de la simulation soit satisfaite)

- Déterminer l'événement le plus imminent et le retirer de la liste des événements (supposons que cet événement est de type i et se produit à l'instant t)
- $\text{clock} = t$
- $sA = sA + (\text{clock} - \text{last_event}) \cdot n$
- $\text{last_event} = \text{clock}$
- Appeler la routine d'événement correspondant au type i

Modèle de file d'attente à serveur unique (3a / 4)

■ Événement d'arrivée – type 1

- Déterminer inter_t, le temps inter-arrivée entre les arrivées actuelle et suivante
- Planifier un événement d'arrivée à clock + inter_t
- $n = n + 1$
- Ajouter le client arrivé à la fin de la file d'attente, et enregistrer son heure d'arrivée (donnée par clock)
- Si status est idle, appeler la routine pour l'événement début de service

Modèle de file d'attente à serveur unique (3b / 4)

■ Événement de début de service – type 2

- Retirer le client du début de la file d'attente, et récupérer son heure d'arrivée ($t_{arrival}$)
- $nW = nW + 1$
- $sW = sW + (\text{clock} - t_{arrival})$
- Si $nW = m$ (condition d'arrêt de la simulation), quitter la boucle principale
- $\text{status} = \text{busy}$
- Déterminer serv_t , le temps de service du client
- Planifier un événement de départ à $\text{clock} + \text{serv_t}$

Modèle de file d'attente à serveur unique (3c / 4)

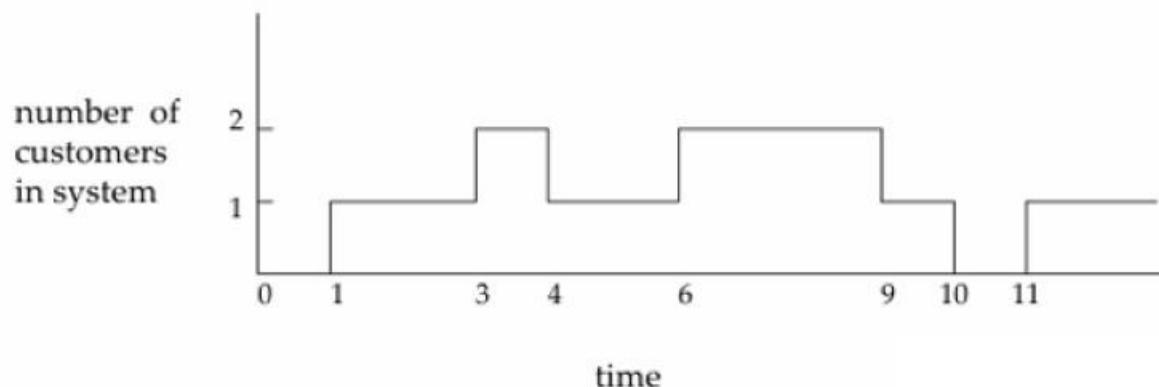
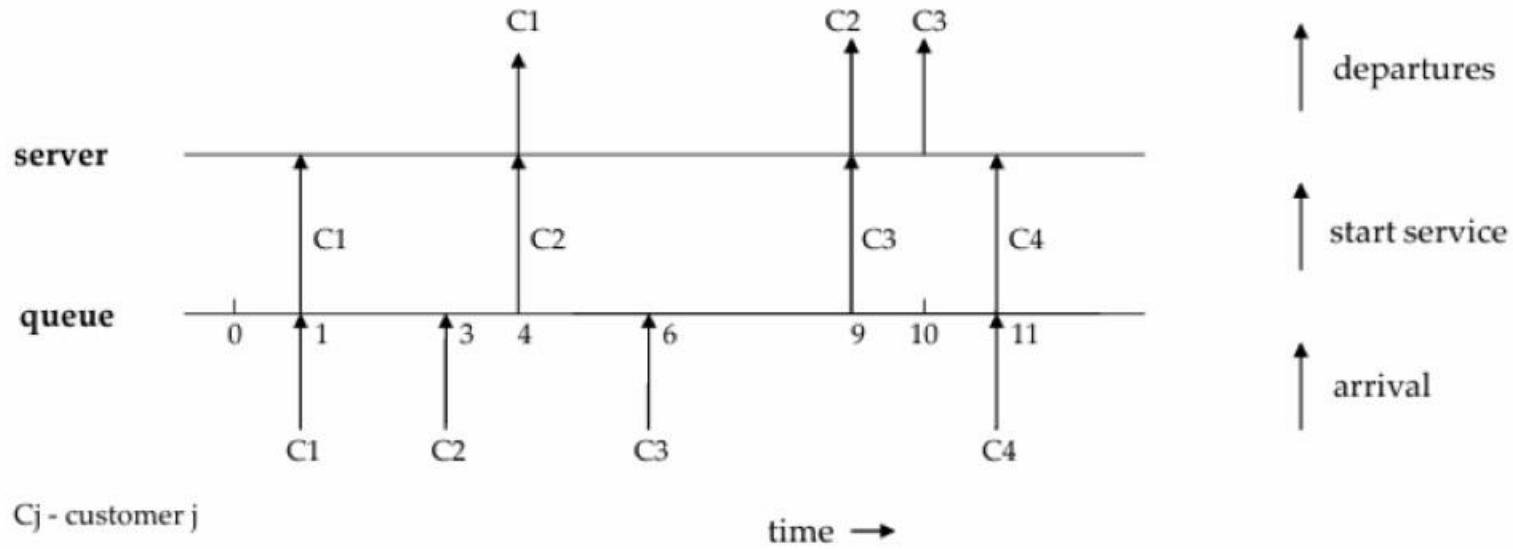
■ Événement de départ – type 3

- $n = n - 1$
- status = idle
- Si $n > 0$, appeler la routine pour l'événement début de service

■ Générateur de rapport

- Temps d'attente moyen : $\bar{W} = sW/nW$
- Nombre moyen de clients dans le système : $\bar{n} = sA/clock$
- Afficher les résultats

Séquence d'événements

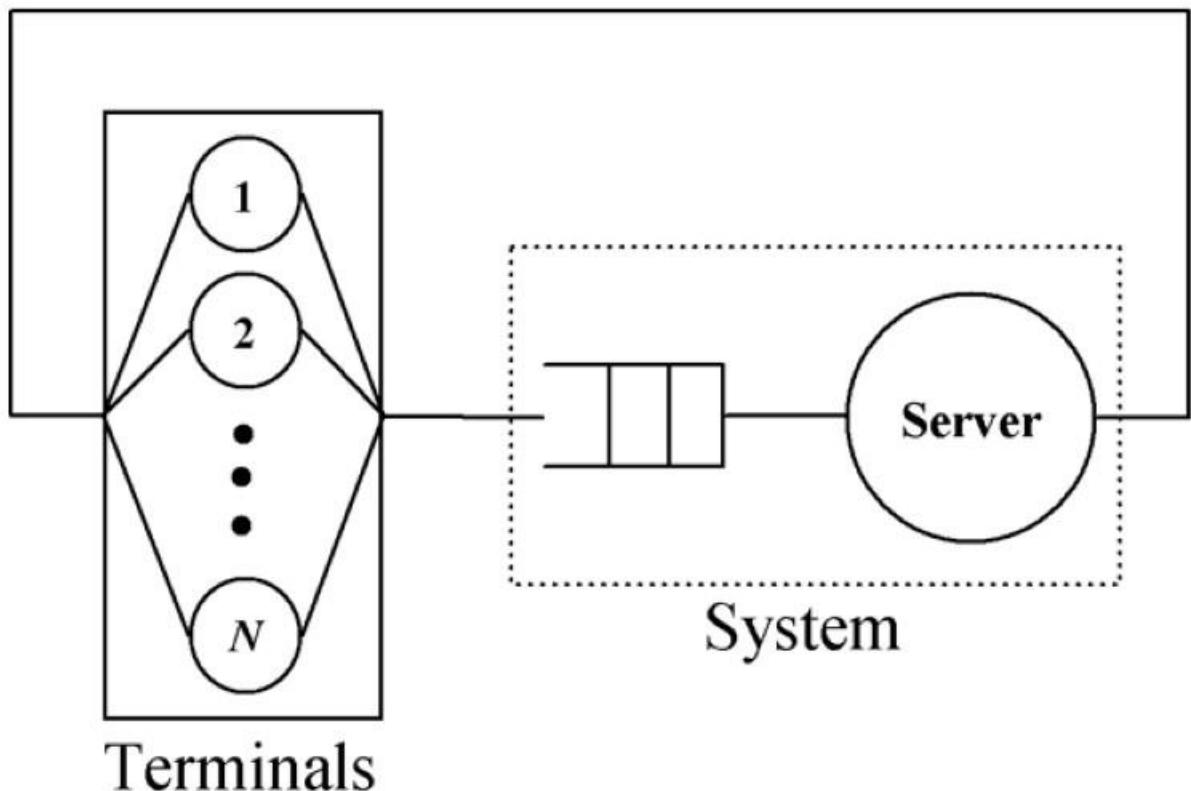


Exemple Modèle de file d'attente à serveur unique

Horloge	Événement	État	n (clients dans le système)	Liste des événements	File d'attente	nw	sw
0	—	inactif	0	(A, 1)		0	0
1	A	occupé	1	(A, 3), (D, 4)		1	0
3	A	occupé	2	(D, 4), (A, 6)	(C2, 3)	1	0
4	D	occupé	1	(A, 6), (D, 9)		2	1
6	A	occupé	2	(D, 9), (A, 11)	(C3, 6)	2	1
9	D	occupé	1	(D, 10), (A, 11)		3	4
10	D	inactif	0	(A, 11)		3	4
11	A	occupé	1	(A, 15), (D, 17)		4	4

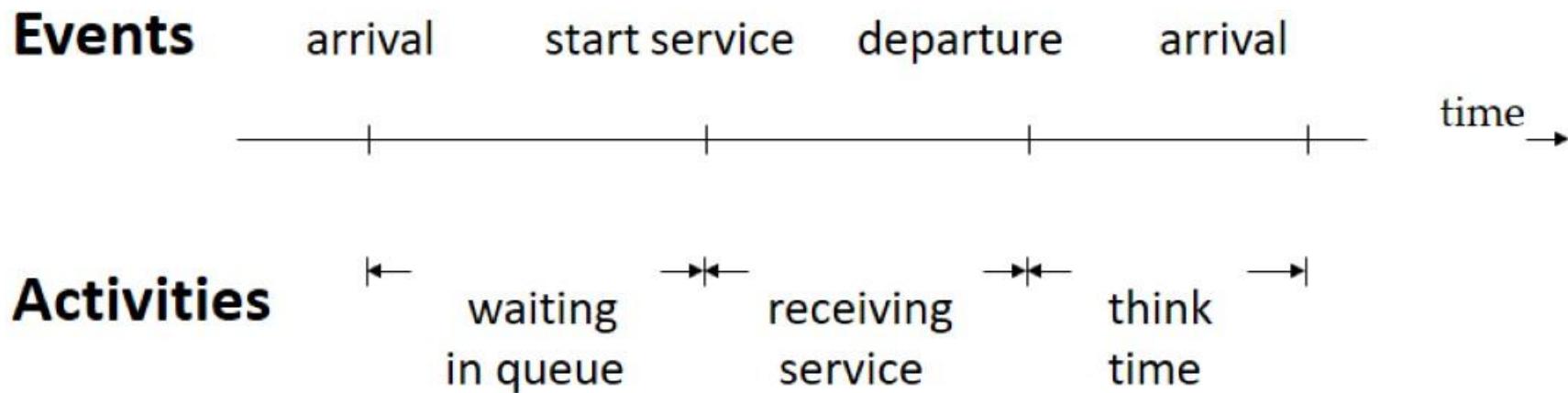
Modèle de file d'attente à serveur unique

■ Modèle à population finie



- Le nombre d'utilisateurs n'est pas élevé.
- Le comportement de chaque utilisateur est modélisé explicitement, notamment en ce qui concerne le schéma d'arrivée.
- Le taux d'arrivée dépend de l'état du système.
- Définition : Temps de réflexion (Think time) : temps écoulé entre la fin de la requête précédente et la soumission de la requête suivante.

Modèle à population finie



Modèle à population finie

- Hypothèses

- Les temps de service sont i.i.d. (indépendants et identiquement distribués) et indépendants de l'état du système
- Les temps de réflexion (think times) sont i.i.d. et indépendants de l'état du système
- Ordonnancement (premier arrivé, premier servi)
- Le système est vide à l'instant zéro
- Pour chacun des N utilisateurs, la première requête est soumise après un temps de réflexion
- Les arrivées suivantes dépendent des fins de service précédentes
- La simulation s'arrête à l'instant `term_sim`

Modèle à population finie

- Initialisation

- $\text{clock} = 0$
- $\text{status} = \text{inactif}$
- $n = 0$
- Initialiser la file d'attente comme vide
- Initialiser la liste des événements comme vide
- Pour chaque utilisateur j ($j = 1$ à N) :

Déterminer think_t , le temps de réflexion de l'utilisateur j

Programmer un événement d'arrivée à l'instant $\text{clock} + \text{think_t}$

Fin pour

- Programmer un événement de fin de simulation à l'instant term_sim

Modèle à population finie

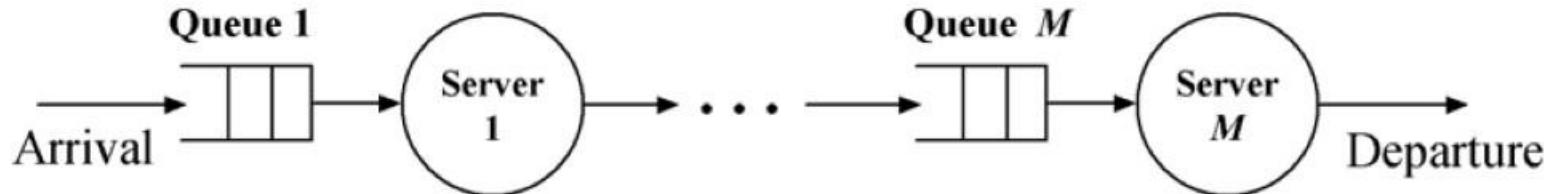
- Événement d'arrivée
 - $n = n + 1$
 - Insérer le client arrivant à la fin de la file d'attente
 - Si status est inactif, appeler la routine de début de service

- Événement de début de service
 - Retirer le client en tête de la file d'attente
 - status = occupé
 - Déterminer serv_t, le temps de service du client
 - Programmer un événement de départ à l'instant clock + serv_t

Modèle à population finie

- Événement de départ
 - $n = n - 1$
 - status = inactif
 - Déterminer think_t (temps de réflexion)
 - Programmer un événement d'arrivée à l'instant clock + think_t
 - Si $n > 0$, appeler la routine de l'événement début de service
- Événement de fin de simulation
 - Quitter la boucle principale

Exemple: Tandem Queue – M étapes

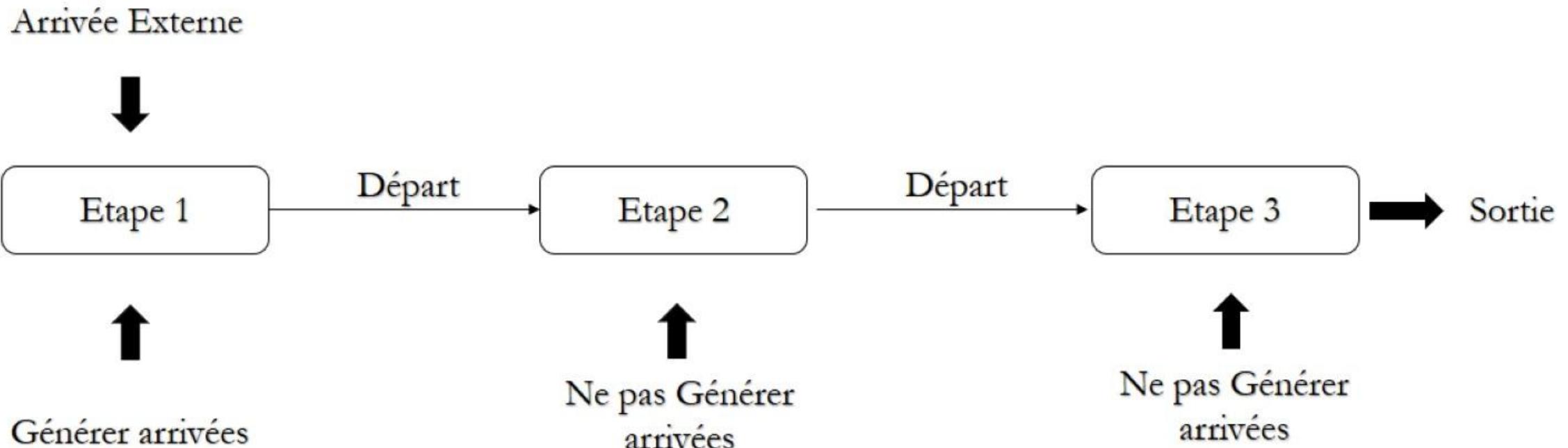


- Sous-systèmes et interactions
 - M sous-systèmes : un pour chaque étape
 - Un départ de l'étape i devient une arrivée à l'étape $i+1$ ($i = 1 \text{ to } M-1$)

Programme de simulation

- Utiliser les routines d'événements du modèle de file d'attente à serveur unique pour chacune des M étapes
- Modifications pour implémenter une file en tandem :
 - Événement de départ : pour l'étape i ($i = 1$ à $M-1$)
 - Ajouter l'étape suivante :
 - Invoquer la routine d'événement d'arrivée à l'étape $i+1$
 - Événement d'arrivée : pour l'étape i ($i = 2$ à M)
 - Ne pas planifier le prochain événement d'arrivée !

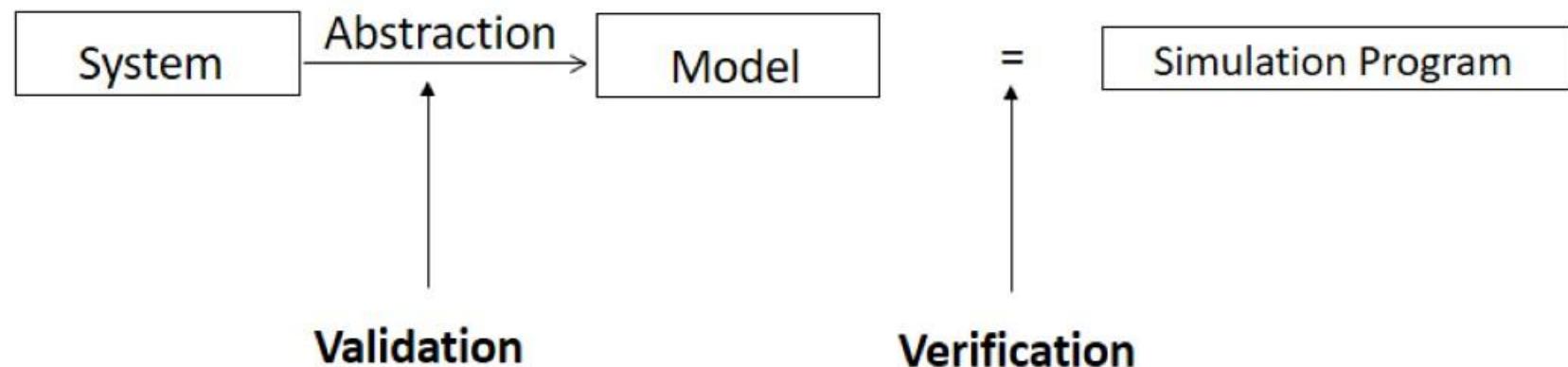
Programme de simulation



Outline

- Concepts in discrete-event simulation
 - Terminology and concepts
 - Two pedagogical examples
- Components of discrete-event simulation
 - Time advance approaches
 - Event scheduling approach
- Manual simulation
 - Grocery store example
- Simulation program
 - Simulation of queuing systems
 - Infinite and finite population model
 - Tandem queue
- Verification and validation of simulation models

Vérification et Validation



Vérification

- Augmenter le niveau de confiance dans la correction du programme de simulation
- Approches
 - Utiliser une « trace » pour déboguer le programme de simulation
La trace est obtenue en affichant les variables d'état, les compteurs statistiques, etc., après chaque événement
 - Vérifier la sortie de la simulation à l'aide de résultats analytiques

Résultats fondamentaux

- Utiliser les résultats fondamentaux des systèmes de files d'attente
- Exemples
 - Pour tout sous-système, le taux moyen d'arrivée, le nombre moyen dans le système et le temps de réponse moyen doivent être cohérents avec la formule de Little.

Résultats analytiques

- Vérifier les résultats pour les cas où des résultats analytiques sont connus
- Exemples
 - Réseaux ouverts avec une distribution exponentielle des temps inter-arrivée et une distribution uniforme des temps de service.
 - Exécuter la simulation pour le cas d'une distribution exponentielle des temps de service.
 - Vérifier si les résultats de la simulation sont cohérents avec les résultats analytiques connus.

Validation

- Le modèle doit être « suffisamment bon » (subjectif).
- Recueillir l'avis d'experts sur les composants du système qui doivent être modélisés avec soin, par exemple, le goulot d'étranglement.
- Un modèle doit être valide pour les mesures de performance (visées).
- Le modèle le plus valide n'est pas nécessairement le modèle le plus rentable (ou le plus économique)

Approche en trois étapes pour la validation

1. Construire un modèle doté d'une forte validité faciale

- Doit paraître raisonnable aux personnes qui connaissent bien le système modélisé.

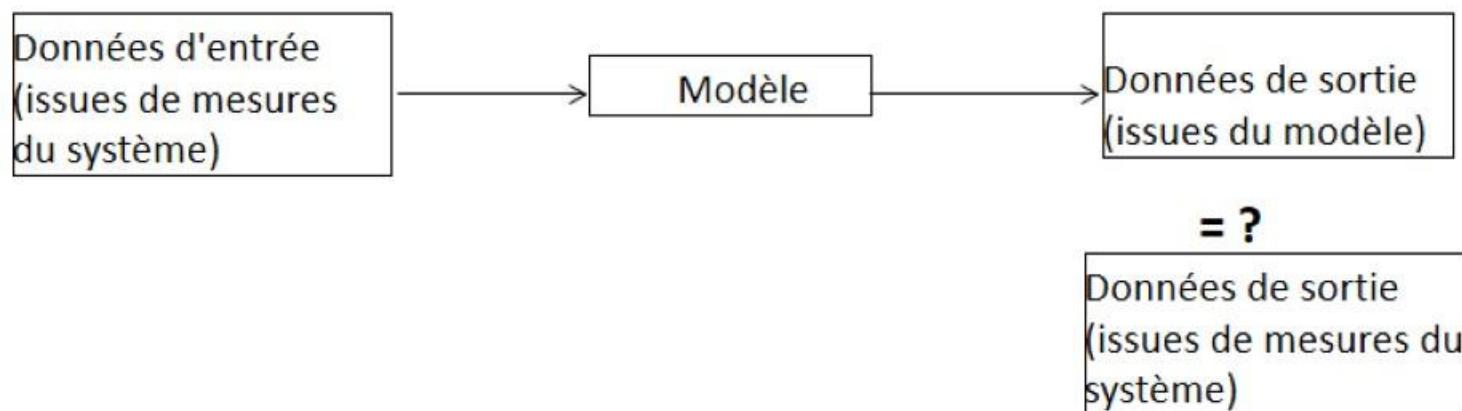
2. Validation des hypothèses du modèle

- Hypothèses structurelles : entités, attributs, ensembles, etc.
- Hypothèses sur les données
 - Collect reliable data
 - Identify appropriate distribution
 - Validate the assumed distribution

Approche en trois étapes pour la validation

3. Validation de la relation entrée-sortie

- Le modèle doit être capable de prédire le comportement du système dans les conditions existantes



Cela peut être fait à l'aide de données historiques collectées à des fins de validation.