



Politecnico  
di Torino



e-Lite



# Discrete Event Simulation

Tecniche di Programmazione – A.A. 2021/2022

# Strategy

---

- ▶ Decision makers need to evaluate beforehand the impact of a strategic or tactical move
  - ▶ But some process are just “too complex”
    - ▶ Mathematical models is too abstract
    - ▶ Building real systems with multiple configurations is too expensive
- ⇒ Simulation is a good compromise



# Simulation

---

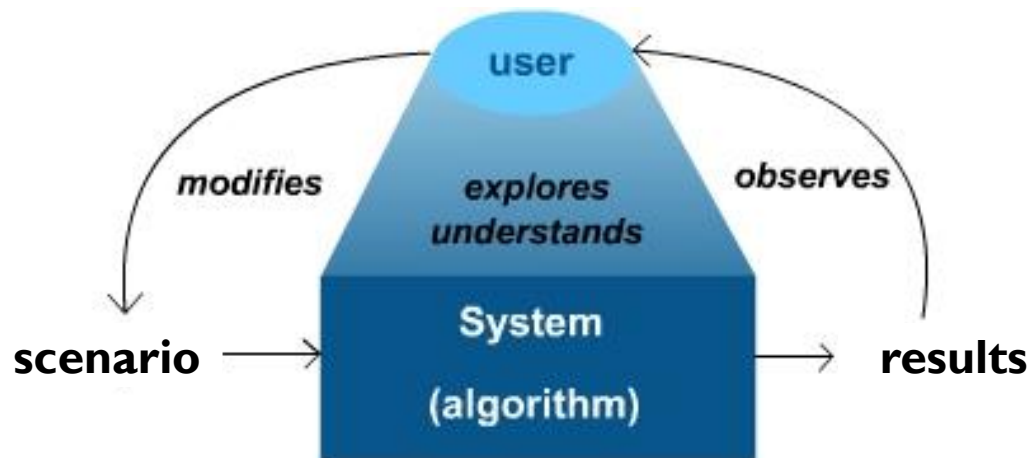
*Simulation is the process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behavior of the system or of evaluating various strategies (within the limits imposed by a criterion or set of criteria) for the operation of a system*

– Shannon

# What-if analysis

---

- ▶ A data-intensive simulation whose goal is to inspect the behavior of a complex system under some given hypotheses (called “scenarios”)
- ▶ What-if analysis  $\neq$  Forecasting



# Disadvantages

---

- ▶ Simulation can be expensive and time consuming
- ▶ Each model is unique
- ▶ Managers must choose solutions they want to try in scenarios
- ▶ Overfitting vs. non-repeatability

# Simulation tools

---

- ▶ **Spreadsheets**

- ▶ Excel
- ▶ Calc
- ▶ Numbers

- ▶ **Ad-hoc**

- ▶ Applix TMI
- ▶ Powersim
- ▶ QlikView
- ▶ SAP BPS
- ▶ SAS Forecast S.
- ▶ ...

# Simulation tools

---

- ▶ **Write your own simulator!**
  - ▶ from scratch
  - ▶ in Java



# Taxonomy

---

- ▶ **Deterministic or Stochastic**
  - ▶ Does the model contain stochastic components?
- ▶ **Static or Dynamic**
  - ▶ Is time a significant variable?
- ▶ **Continuous or Discrete**
  - ▶ Does the system state evolve continuously or only at discrete points in time?



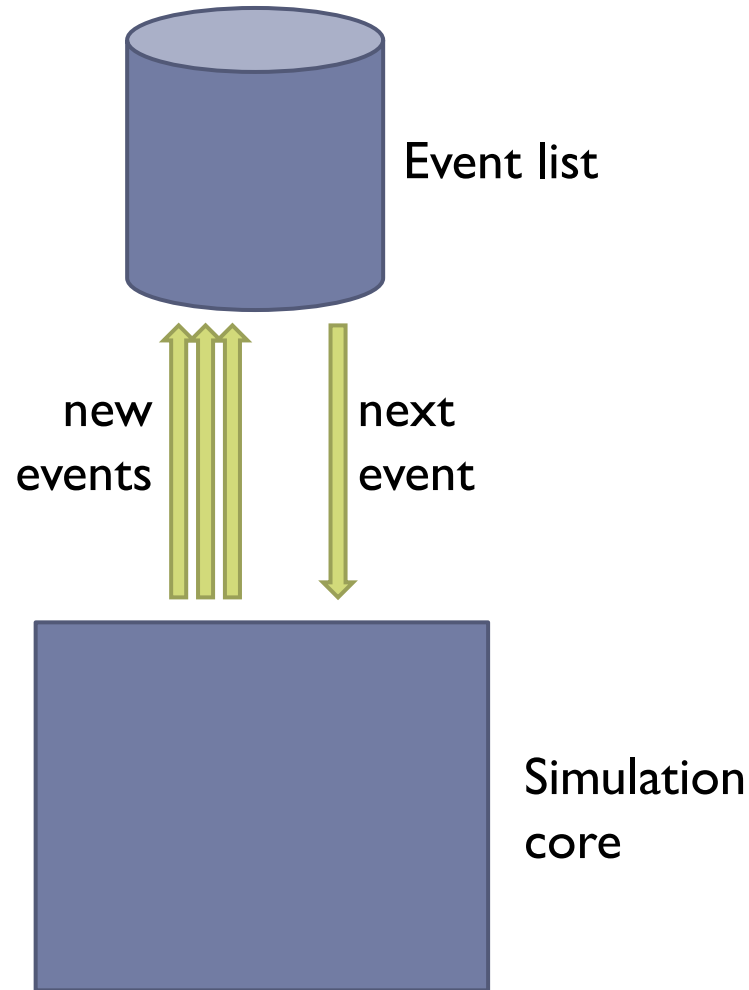
# Discrete Event Simulation (DES)

---

- ▶ Discrete event simulation is **dynamic** and **discrete**
- ▶ It can be either deterministic or stochastic
- ▶ Changes in state of the model occur at discrete points in time
- ▶ The model maintains a list of events (“*event list*”)
  - ▶ At each step, the scheduled event with the lowest time gets processed (i.e., the event list is a *priority queue*)
  - ▶ The event is processed, new events are scheduled

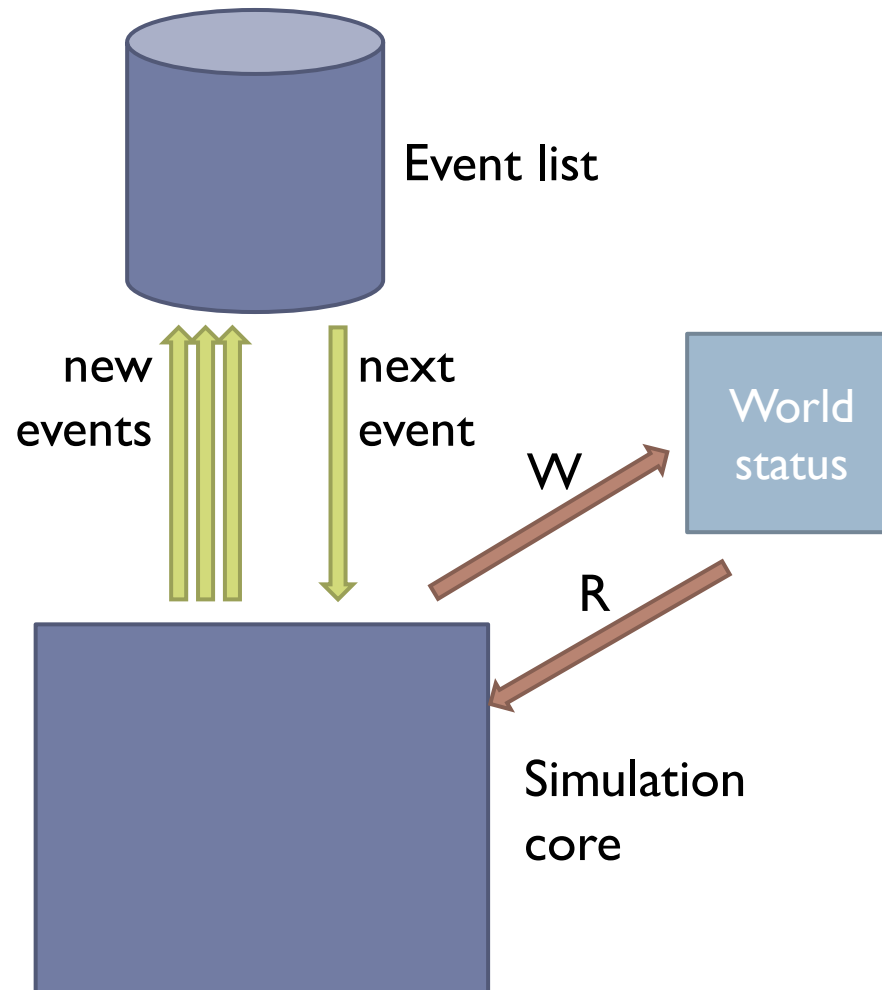
# Discrete Event Simulation (DES)

---



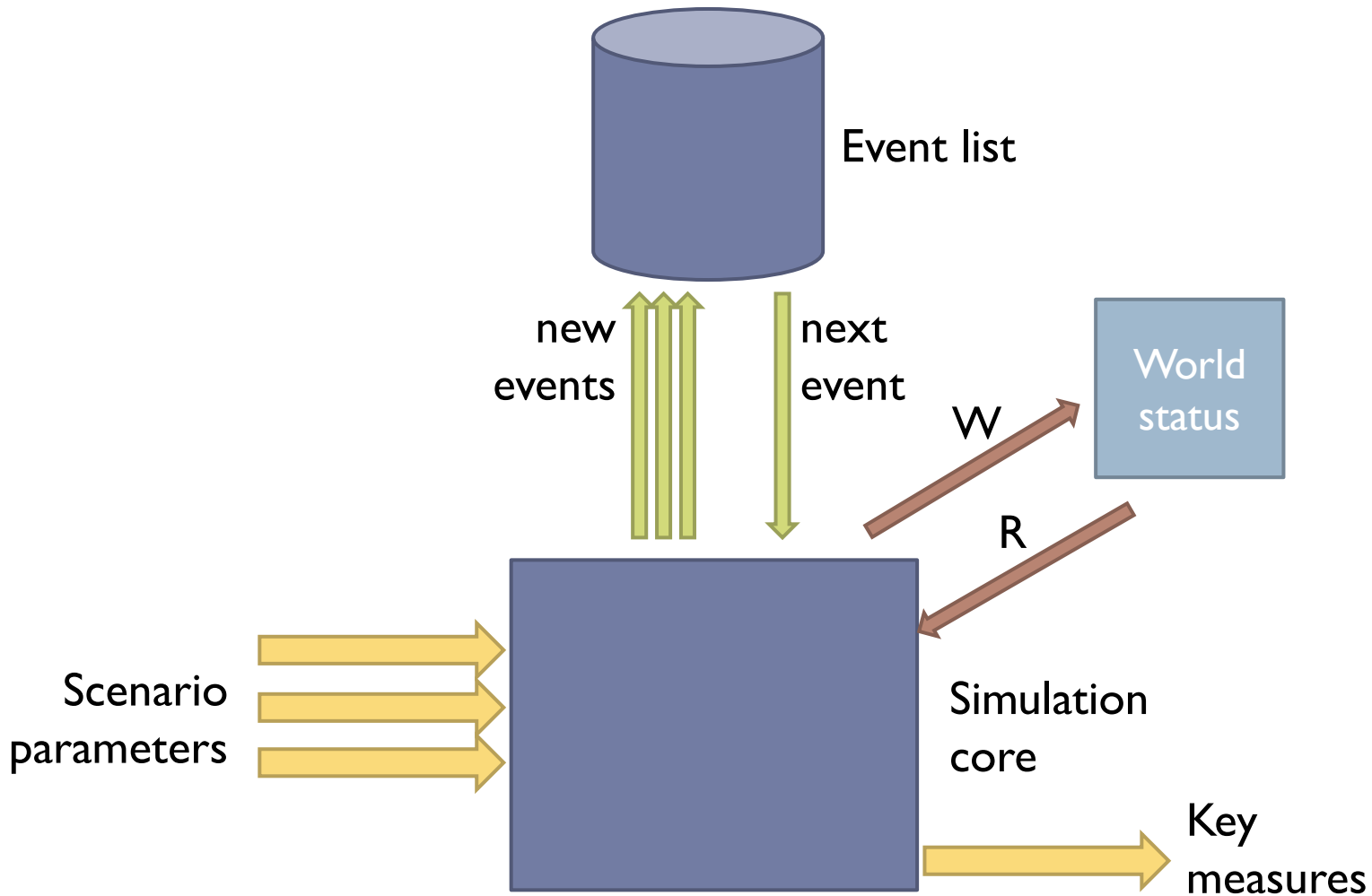
# Discrete Event Simulation (DES)

---



# Discrete Event Simulation (DES)

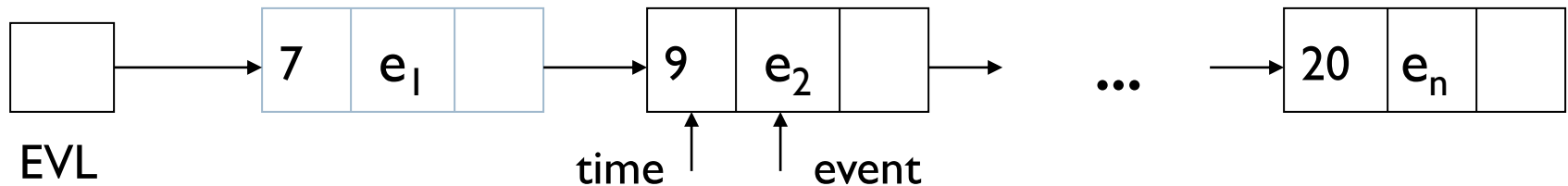
---



# The event list

---

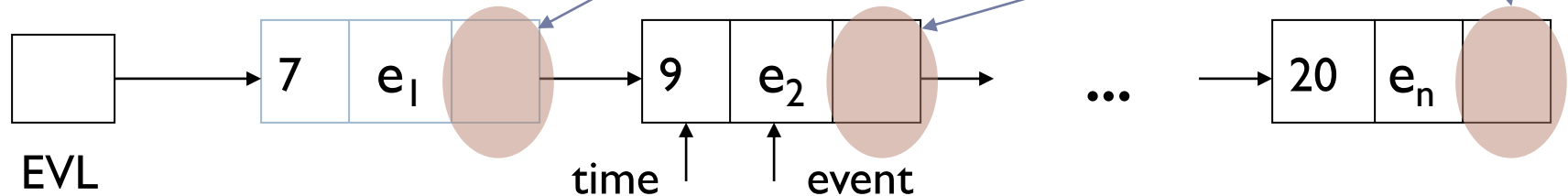
- ▶ An event contains at least two fields of information
  - ▶ time of occurrence (timestamp): time when the event should happen (in the “simulated future”)
  - ▶ what the event represents (a **type** attribute, or a subclass)



- ▶ Simulation terminates when the event list is empty
- ▶ Conceptually endless simulations, like weather, terminate at some arbitrary time

# The event list

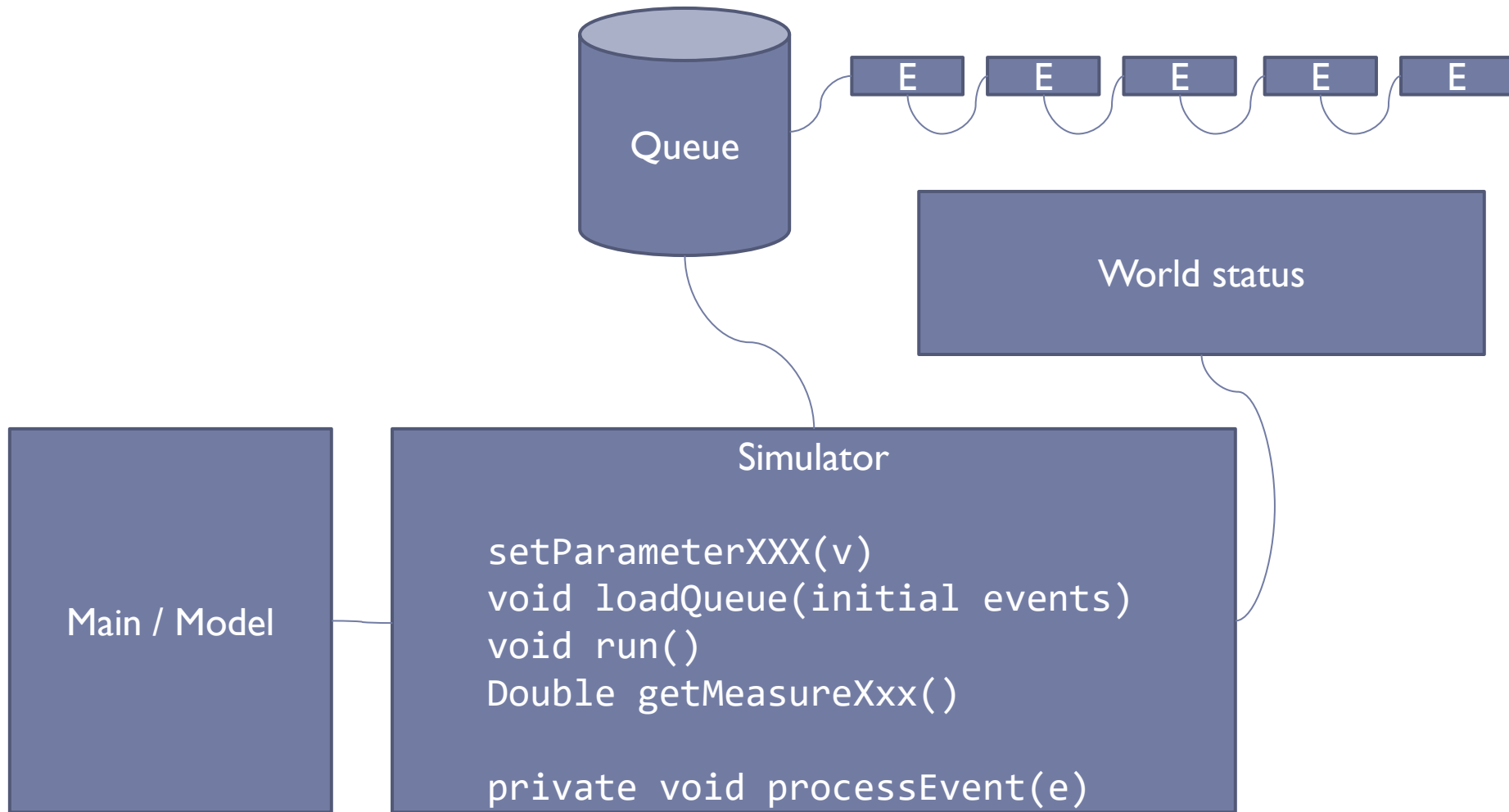
- ▶ An event contains at least two fields of information
  - ▶ time of occurrence (timestamp): time when the event will happen (in the “simulated future”)
  - ▶ what the event represents (a **type** and **id**)



- ▶ Simulation terminates when the event list is empty
- ▶ Conceptually endless simulations, like weather, terminate at some arbitrary time

# Simulator architecture

---



# World Status

---

- ▶ A set of variables / collections / graphs / ... that represent the current state (the *present*) of the simulation
- ▶ The simulation makes the world status evolve, through a series of events
- ▶ The world status may influence / constrain how the events are processed
- ▶ The world status includes the measures of interest.



# General behavior: Simulator

---

- ▶ `setParameterXXX`: defines the simulation strategy and parameters, and initializes the World Status
  - ▶ Can also be in the Simulator constructor
- ▶ `loadQueue`: defines the initial contents of the simulation queue, *at time zero*
- ▶ `run`: executes the simulation loop
  - ▶ extract an event from the queue
  - ▶ `processEvent(e)`
- ▶ `getMeasureXXX`: allows to access the results of the simulated variables, after the completion of the loop

# processEvent(e)

---

- ▶ Analyzes `e.getType()`
- ▶ Depending on:
  - ▶ The simulation parameters (constants) and strategy
  - ▶ The type of event
  - ▶ The value(s) associated with the event
  - ▶ The current world status
- ▶ It performs actions:
  - ▶ (Optional) updates the current world status
  - ▶ (Optional) generates and inserts new events (in the future)
  - ▶ (Optional) updates the measures of interest

# Handling time

---

## Synchronous

- ▶ Events (all/most/some) correspond to the passing of time
  - ▶ Easy to generate systematically (all at the beginning, or each one generates the next)
- ▶ When a new day/hour/months ticks, something needs to be done
- ▶ May be intermixed by other events, at arbitrary times

## Asynchronous

- ▶ Something happens in the simulated world
- ▶ May happen at any time instant
- ▶ The simulated time will «jump» to the nearest interesting event

# Handling Randomness

---

## Deterministic

- ▶ All actions are purely deterministic (initial events, event processing)
- ▶ Repeating the simulation, with equal parameters, will yield the same result. Always.

## Stochastic

- ▶ Random initial events (times, values, types)
- ▶ Randomness in event processing (eg. in 10% of the cases simulate a fault)
- ▶ Repeating the simulation will yield different measures
- ▶ Simulation should be repeated and the measures should be averaged



# Example 1

Discrete Event Simulation

# Example: Car Sharing

---

- ▶ We want to simulate a deposit of shared cars.
  - ▶ Initially we have  $NC$  cars
- ▶ A new client comes every  $T\_IN$  minutes
  - ▶ If there are available cars, he lends one car, for a duration of  $T\_TRAVEL$  minutes
  - ▶ If there are no cars, he is a dissatisfied client
- ▶ Compute the number of dissatisfied clients, at the end of the day, as a function of  $NC$ .
- ▶  $T\_IN = 10$  minutes
- ▶  $T\_TRAVEL = \text{random} (1 \text{ hour}, 2 \text{ hours}, 3 \text{ hours})$

# Simulator data

---

## Events

- ▶ Client arrives
- ▶ Client returns car

## World model

- ▶ Number of total cars
- ▶ Number of available cars
  
- ▶ Number of clients served
- ▶ Number of dissatisfied clients

# Variants

---

- ▶ Remember “who” is the client, at return time
- ▶ Model different kinds of cars (A, B, C).
  - ▶ A client wants one kind of car, but he may accept a “better” car (cost for the company)
- ▶ Model different car rental locations
  - ▶ A car is taken at location “x” and returned at location “y”





# Example: Emergency

---

- ▶ We simulate the behavior of an **Emergency department** in an hospital.
- ▶ The department is organized in two sections
  - ▶ A single Triage, where patients are received, quickly examined, and assigned a severity code
  - ▶ A number  $NS$  of doctor studios, that operate in parallel. Each doctor will receive the next patient, act on him, and then release him
  - ▶ The severity code gives priority in accessing the doctors. Patients with the same severity, will be called in arrival order.

# Severity codes

---

- ▶ **WHITE**: not urgent, may wait without problems
  - ▶ After `WHITE_TIMEOUT`, if not served, goes home
- ▶ **YELLOW**: serious but not urgent
  - ▶ After `YELLOW_TIMEOUT`, if not served, becomes **RED**
- ▶ **RED**: serious and urgent, risking life, must be served as soon as possible
  - ▶ After `RED_TIMEOUT`, if not served, becomes **BLACK**
- ▶ **BLACK**: dead. No need to be served.

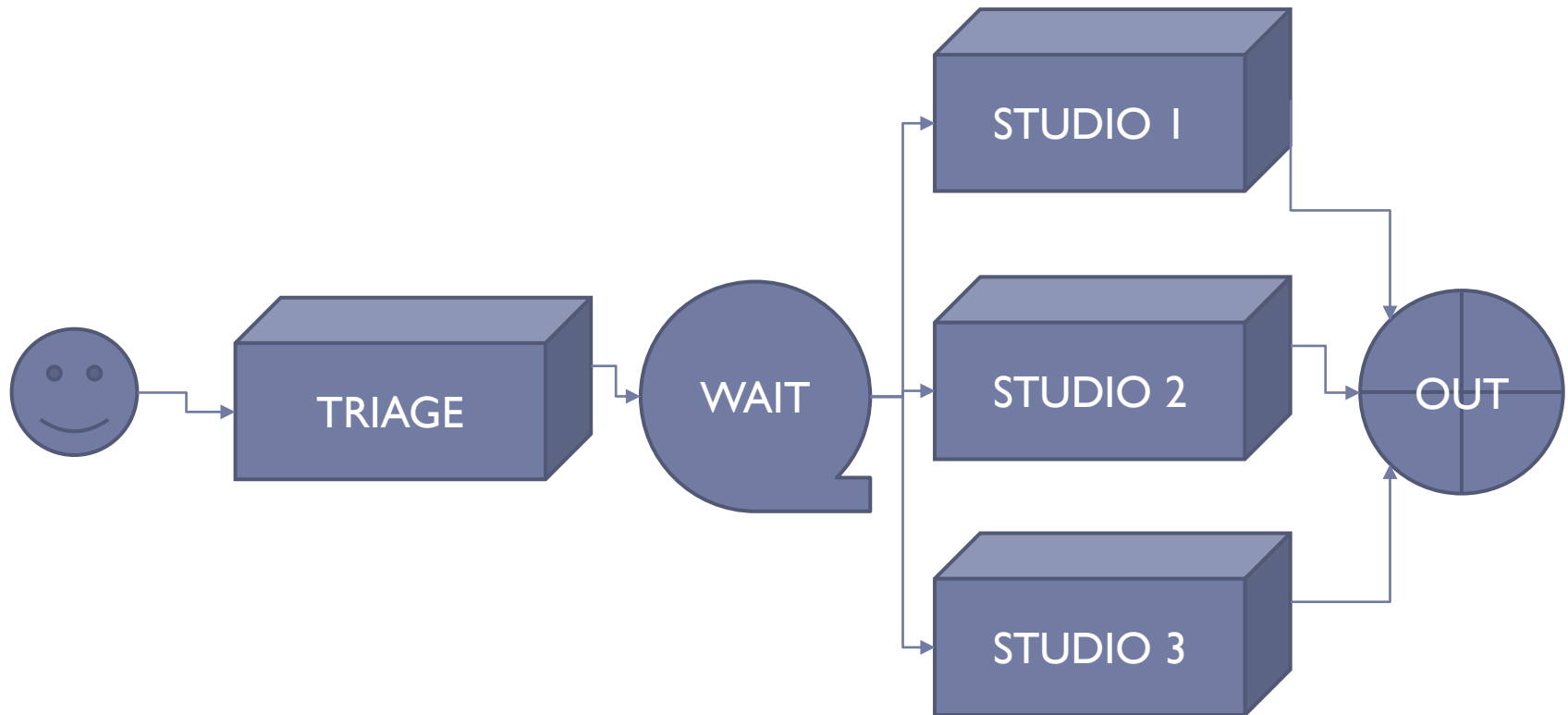
# Timing

---

Phase	Required time	Example
Triage	DURATION_TRIAGE	5 minutes
Handling a White patient	DURATION_WHITE	10 minutes
Handling a Yellow patient	DURATION_YELLOW	15 minutes
Handling a Red patient	DURATION_RED	30 minutes
Handling a Black patient	N/A	not needed

# Patients workflow

---



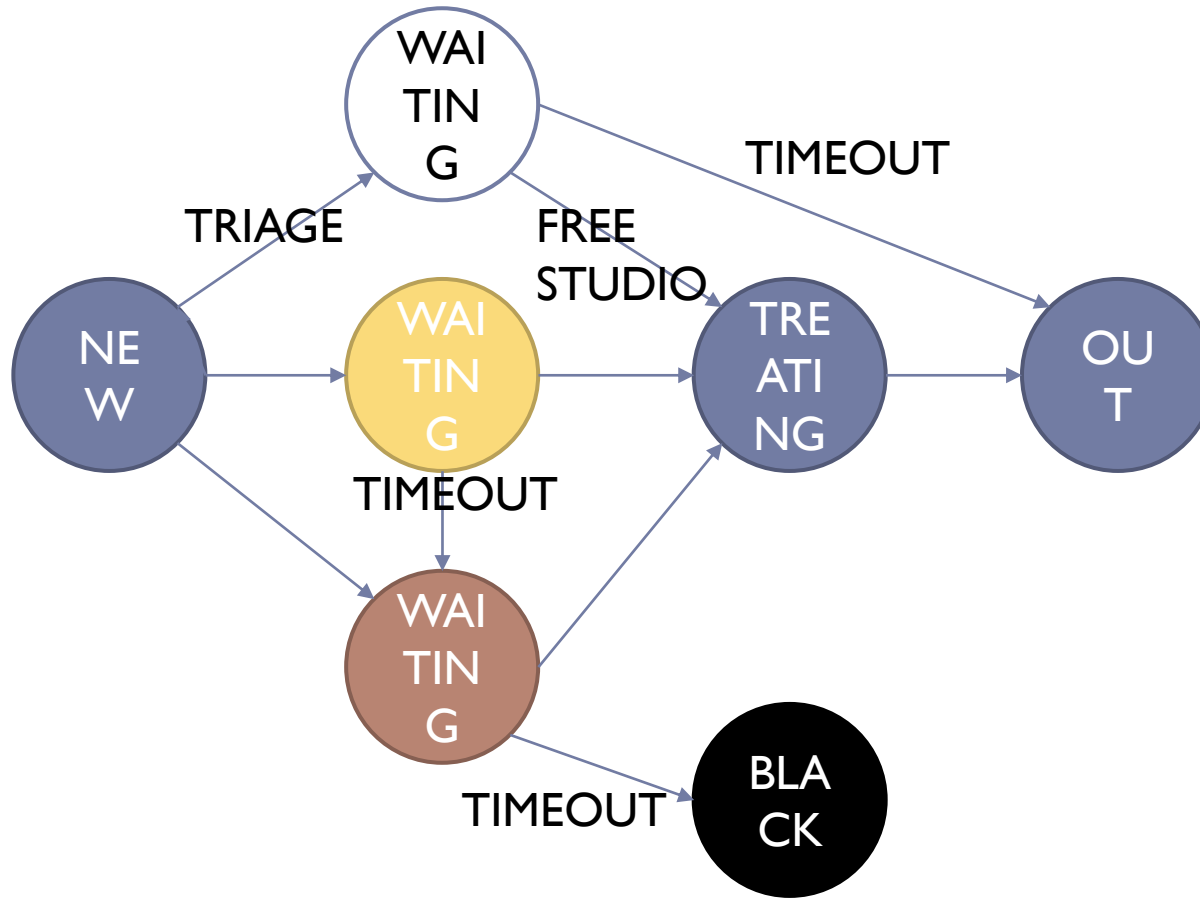
# World Model

---

- ▶ Collection of patients
- ▶ For each patient:
  - ▶ Patient status

# Evolution of patient status

---



# Simulation goals

---

## Input

- ▶ Parameter: NS
- ▶ Initial events:
  - ▶ NP patients
  - ▶ arriving every  $T\_ARRIVAL$  minutes
  - ▶ with a round-robin severity (white/yellow/red/white/...)
- ▶ Simulate from 8:00 to 20:00

## Output

- ▶ Number of patients dismissed
- ▶ Number of patients that abandoned
- ▶ Number of patients dead








# Randomizing

---

- ▶ Input arrival times every  $T\_ARRIVAL \pm \text{random}\%$
- ▶ Input severity probabilities (PROB\_WHITE, PROB\_YELLOW, PROB\_RED)
- ▶ Variable processing time (DURATION\_TRIAGE, DURATION\_WHITE, DURATION\_YELLOW, DURATION\_RED  $\pm \text{random}\%$ )
- ▶ Etc...

# Licenza d'uso



- ▶ Queste diapositive sono distribuite con licenza Creative Commons “Attribuzione - Non commerciale - Condividi allo stesso modo (CC BY-NC-SA)”
- ▶ Sei libero:
  - ▶ di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare quest'opera 
  - ▶ di modificare quest'opera 
- ▶ Alle seguenti condizioni:
  - ▶ **Attribuzione** — Devi attribuire la paternità dell'opera agli autori originali e in modo tale da non suggerire che essi avallino te o il modo in cui tu usi l'opera. 
  - ▶ **Non commerciale** — Non puoi usare quest'opera per fini commerciali. 
  - ▶ **Condividi allo stesso modo** — Se alteri o trasformi quest'opera, o se la usi per crearne un'altra, puoi distribuire l'opera risultante solo con una licenza identica o equivalente a questa. 
- ▶ <http://creativecommons.org/licenses/by-nc-sa/3.0/>