
GEBAK specialiteiten

Universiteit Utrecht

[Faculty of **Science**
Information and Computing Sciences]

Simulation

Lecture 2

Modeling

ADS, lecture 2

In this lecture

- We study a basic example of discrete-event simulation to learn the modeling principles.
- After the lecture, you should be able to make basic simulation models, such as the exercises of Ch 1 in Law.

Discrete-event simulation (discrete, dynamic, stochastic)

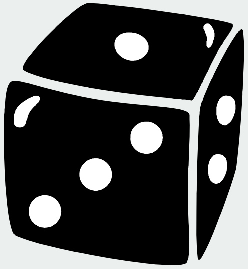
- *State*: collection of variables that describe the system at a particular moment in time
- *Event* may change the state of the system
- *Discrete-event simulation*: state variables change instantaneously at separate points in time, *uncertainty* is included

Single-server queue

- Example: calling the assistant of the family doctor (huisarts), baker’s shop with one employee, help desk

- t_i= arrival time customer i
- $A_i = t_i - t_{i-1}$ inter-arrival time
- S_i= service time customer i
- c_i = departure time customer i
- D_i= waiting time i

- $D_i = \max(0, c_{i-1} - t_i)$=waiting time i
- $c_i = t_i + D_i + S_i = \max(t_i, c_{i-1}) + S_i$
- e_j = time event j
- *S* and *A* are stochastic variables



Single-server queue

- Events:
  - Arrival of customer
  - Departure of customer

Simulation clock

- Next-event time advance
- Fixed increment time advance

Discrete-event simulation always applies next event time advance

Performance measures

- Average waiting time
- Average queue length
- Fraction of time that the server is busy (bezettingsgraad)

Single server queue: performance measures

- Average waiting time:

$$\hat{d}(n) = \frac{\sum_{i=1}^{n} D_i}{n}$$

D_i known at start of service customer i

Single server queue: performance measures (2)

- Average number of customers in queue:

$$\hat{q}(n) = \frac{\sum_{i=0}^{n} iT_i}{T} = \frac{Q(T)}{T}$$

where

  - T_i is time with i customers in queue,
  - $Q(T)$ total waiting time until T,
  - T is total time.

\`calculate surface'

$$\hat{u}(n) = \frac{B(T)}{T}$$

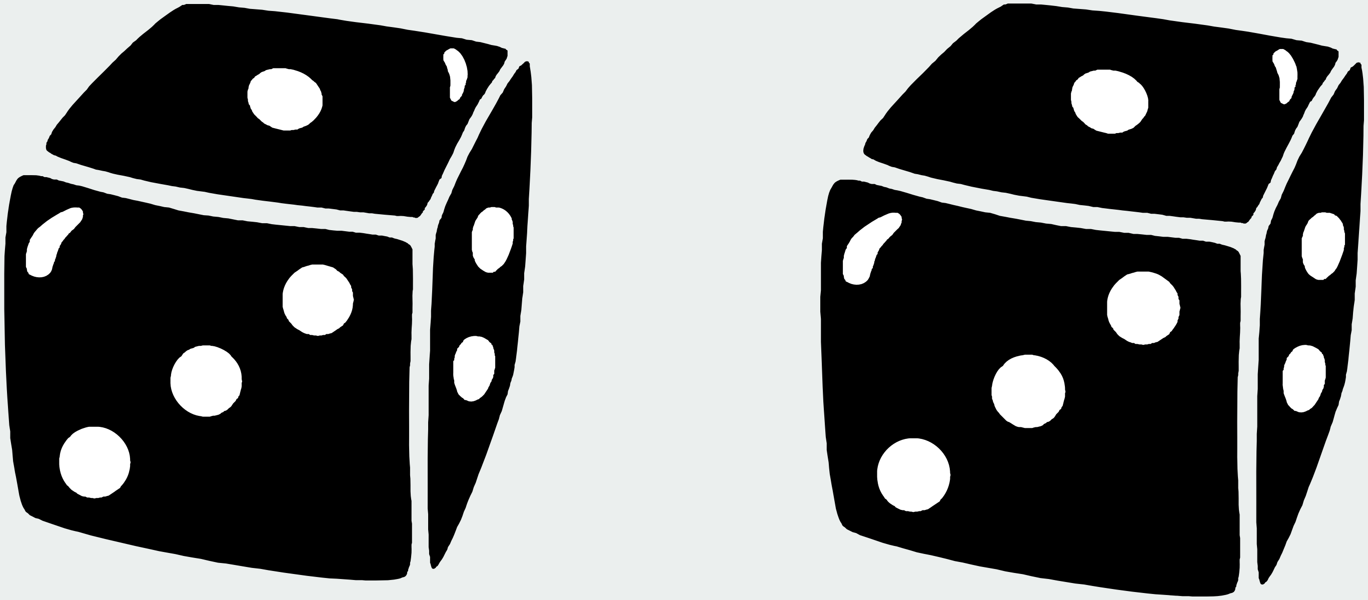
- Fraction of time that server is busy: busy-time/total time,
  - B(T) total busy time until T

\`calculate surface'

State

- Server: idle/busy
- Number of customer is queue
- Arrival times of customers in queue

Simulation by hand



```
while time < runlength
{
   case nextevent of
        arrival:
                time = arrivaltime;
                update statistics busy time B and
                        total queue size Q;
                if server idle
                        then{
                                update delay statistics D and make
                                        server busy (state);
                                schedule new departure;
                        }
                        else add customer to queue (state);
                schedule new arrival;
        departure:
                time = departuretime;
                update statistics busy time B and
                        total queue size Q;
                if queue is not empty
                        then{
                                update delay statistics D start new
                                service (state);
                                schedule new departure;
                        }
                else make server idle (state);
}
```

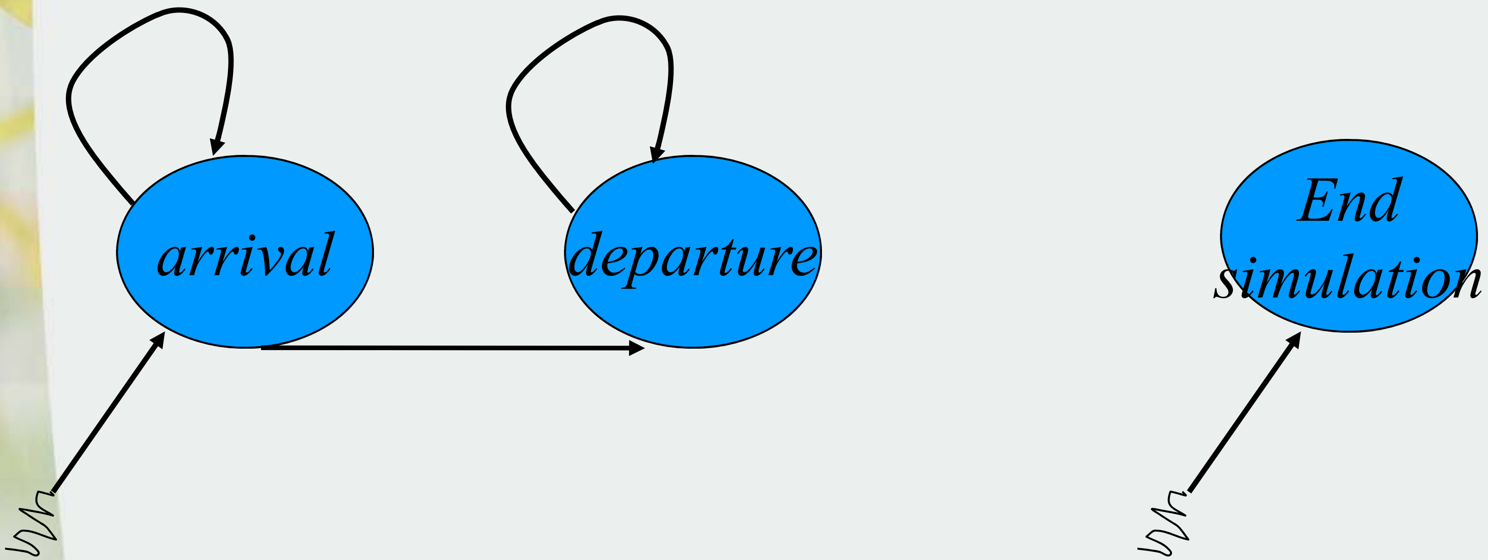
Discrete event simulation:
event-scheduling approach

```
INITIALIZATION

MAIN PROGRAM
while time < runlength
{
        get next event from event list;
        advance simulation time;
        update statistics + system state;
        generate future events and add them to event list;
}
```

Event graph

A → B   Event A may schedule event B

A ⇢ B   Event A may schedule event B with zero delay



⇝ B   Event B is scheduled from initialization of simulation

An event graph is not the same as a workflow diagram.

Development of simulation model

Important to remember!!!

- System description
- Assumptions
- Performance measures
- Events (event graph)
- State
- Event handlers: in words or flow diagram/pseudo-code
  - Update state
  - Update performance measures
  - Generate new events

- Input data/distributions
  - If they are not given, they have to be obtained by a separate input analysis

Bus example

Variation is bad

Bus example

- You want to take a bus. You move to the bus stop without considering the time table.

1. The bus runs 3 times per hour and has inter-arrivaltimes of exactly 20 minutes. You average waiting time equals 10 minutes
2. The bus runs 3 times per hour and has inter-arrivaltimes 10 mins, 30 mins, 10 mins, 30 mins, 10 mins etc.

Now your average waiting time equals:

$$P(arrive\ during\ 30\ mins) * \\ avg.waittime\ given\ that\ you\ arrive\ during\ 30\ minutes + \\ P(arrive\ during\ 10\ mins) * \\ avg.waittime\ given\ that\ you\ arrive\ during\ 10\ minutes = \\ 0.75 * 15 + 0.25 * 5 = 12.5\ mins$$

This is **larger**, since you have a higher probability of arriving during a long inter-arrival-interval!

Queuing system: Analysis

- Utilization factor (bezettingsgraad):

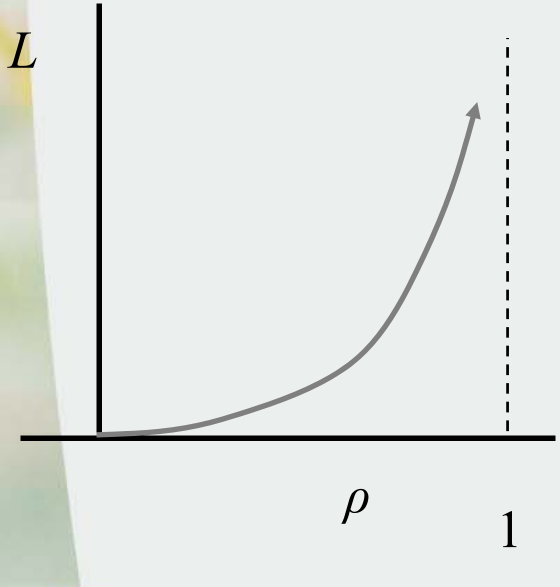
$$\rho = \frac{E(S)}{E(A)} = \lambda E(S)$$

  where
  - *A* inter arrival time,
  - *S* service time,
  - λ arrival intensity

- If inter arrival times are exponentially distributed (negexp): **Poisson process** with intensity λ = 1/E(A)

Queuing system: analysis (2)

- 1 queue, 1 server
- *A, S* exponential distribution (=high variance)
- *L:* Long-term **average** number of clients in the system



$$L = \frac{\rho}{1-\rho}$$

High variation:

Plan more careful !!

Exercise 1.22

- *m* machines:
  - Break down after negexp(8 hours)
  - Repair time negexp(2 hours)
- *s* repair man
  - are assigned in FIFO order to machines
- Costs:
  - Repair man: 10 EURO per hour regardless if they are working
  - Downtime: 50 EURO per machine per hour
- By the simulation we want to find out:
  - How many repairmen do we need, i.e. what should *s* be?
- Question: make a simulation model?
  - Events
  - Event graph
  - Performance measures
  - State
  - Event handlers

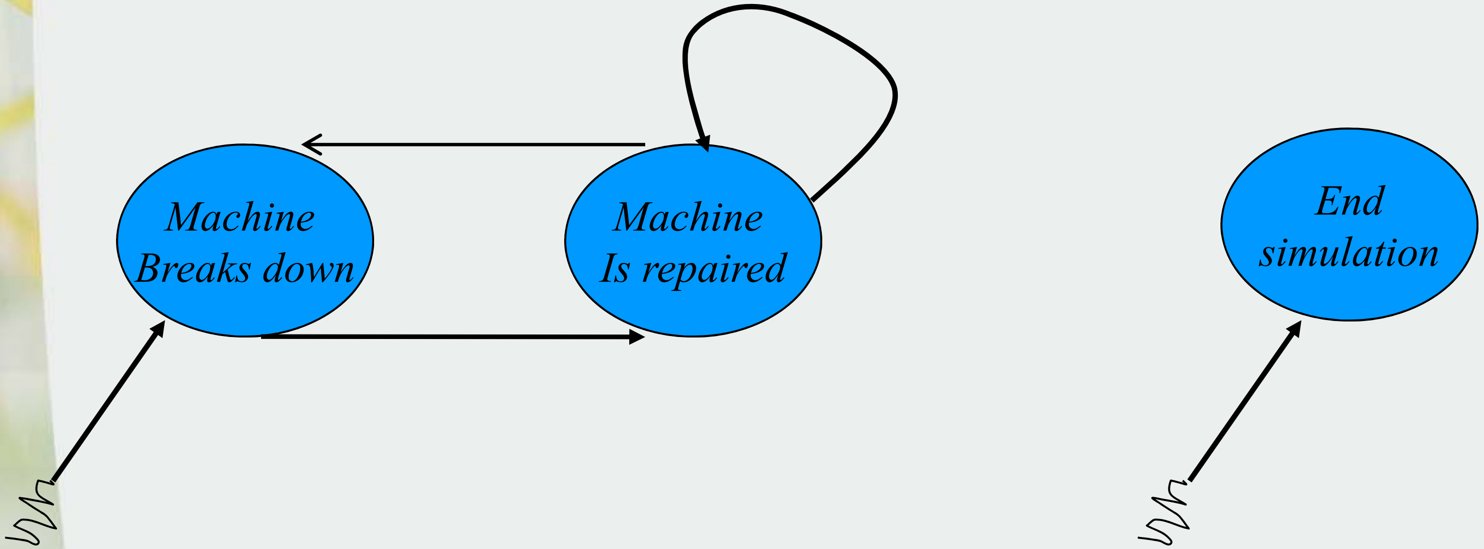
Exercise 1.22 (2)

- Events:
  - Machine breaks down
  - Machine is repaired
  - End of simulation
- State:
  - n_{md} number of machines that are down
  - n_{ir} number of idle repairmen
  - q: length of the machine queue
- Performance measure: total cost
  - Cost of repairman: #simulated hours*10*s
  - Down time cost= 50*(total down time)
  - Total down time is computed as $\sum_{i=0}^{m} i\, T_i^{down}$,
    where T_i^{down} time with *i* machines down.

Exercise 1.22: Event graph

A ⟶ B Event A may schedule event B

A - - - → B Event A may schedule event B with zero delay



 B Event B is scheduled from initialization of simulation

Discrete event simulation:
event-scheduling approach

```
INITIALIZATION

MAIN PROGRAM
while time < runlength
{
        get next event from event list;
        advance simulation time;
        update statistics + system state;
        generate future events and add them to event list;
}
```

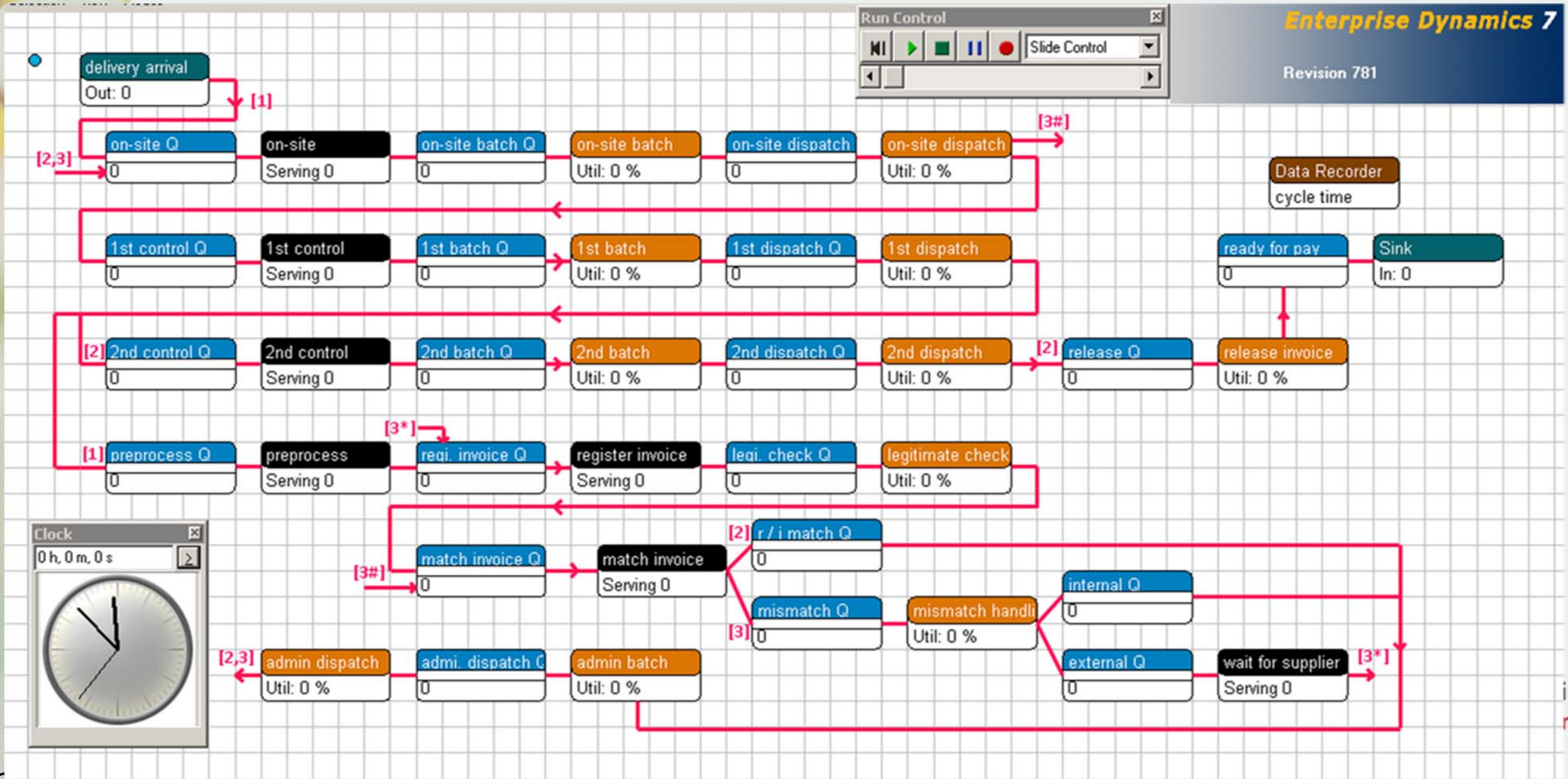
```
while time < runlength
{
    case nextevent of
        breakdown:
                time = breakdowntime;
                update statistics total down time: add $n_{md}(t_{now}-t_{prev})$;
                $n_{md}$ = $n_{md}$ + 1;
                if $n_{ir}$ > 0 (idle repairman available)
                        then{   start repair $n_{ir}$ = $n_{ir}$ - 1;

                                schedule new machinerepaired
                                                        (exp(2h));
                        }
                        else add machine to queue q = q+1;

        machinerepaired:
                time = repairtime;
                update  statistics total down time: add $n_{md}(t_{now}-t_{prev})$;
                $n_{md}$ = $n_{md}$ - 1;
                if q > 0 (queue is not empty)
                        then{   start new repair;
                                q = q-1;
                                schedule new machinerepaired
                                                (exp(2h));
                        }
                        else{ $n_{ir}$ = $n_{ir}$ + 1};
                schedule new breakdown after exp(8h);

}
```

Wrap up/homework

- Now, you should be able to make basic simulation models, such as the exercises of Ch 1 in Law (see website).
- Note, for these exercises you have to write down a simulation model, it is not necessary to program and run the simulation
  - Performance measures
  - Events (event graph)
  - State
  - Event handlers: in words or flow diagram/pseudo-code
    - Update state
    - Update performance measures
    - Generate new events

Implementation

- Programming language
- Simulation package



ience
nces]

Comparison

Programming language

- Event-scheduling
- More detail
- Flexibility
- No hidden functionality
- Faster wrt computation time

Simulation tool

- process (interaction) approach,
- less programming,
- Quick for a simple model
- Graphical options,
- Less debugging,
- Learning curve,