Business Process Simulation

Lecture 3

Laura Genga

Department of Industrial Engineering and Innovation Sciences, Information Systems Group

Overview on lecture modules

a) Conceptual model: main elements
b) Simple queuing system: process and information model
c) Simple queuing system: transition specifications
d) How does simulation work?
e) Petrol station example and manual simulation
f) Additional examples

Overview of today's lecture

Recap: Step 1 and step 2
Step 3: Conceptual model
Manual simulation

Recap Simulation Methodology (7 steps)

STEP 1: Project definition

STEP 2: Design the simulation study

STEP 3: Conceptual model

STEP 4: Executable model and verification

STEP 5: Validation

STEP 6: Experiments and output analysis

STEP 6: Conclusion

TU/e

Recap Simulation Methodology

STEP 1: Project Definition

1.1 Decision frame

1.2 Research questions

1.3 Scope and level of detail

STEP 2: Design the study (black box & assumptions)

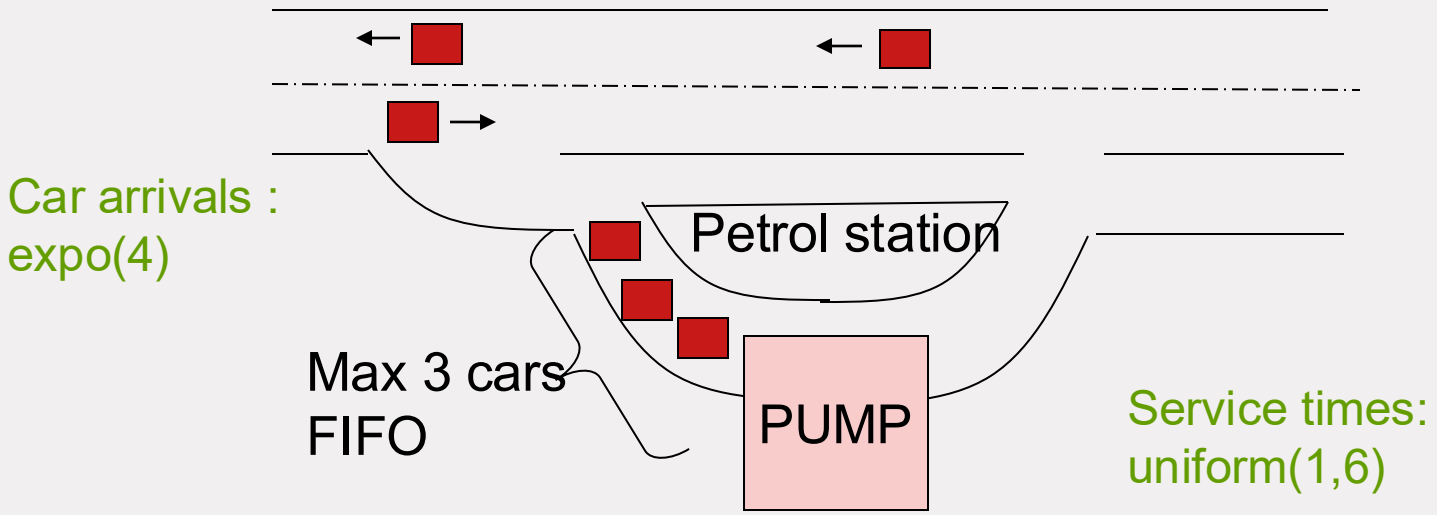
2.1 Black box representation

2.2 Assumptions and givens

2.3 Simulation suitable / needed?

2.4 Number of models

EXAMPLE: The Petrol Station

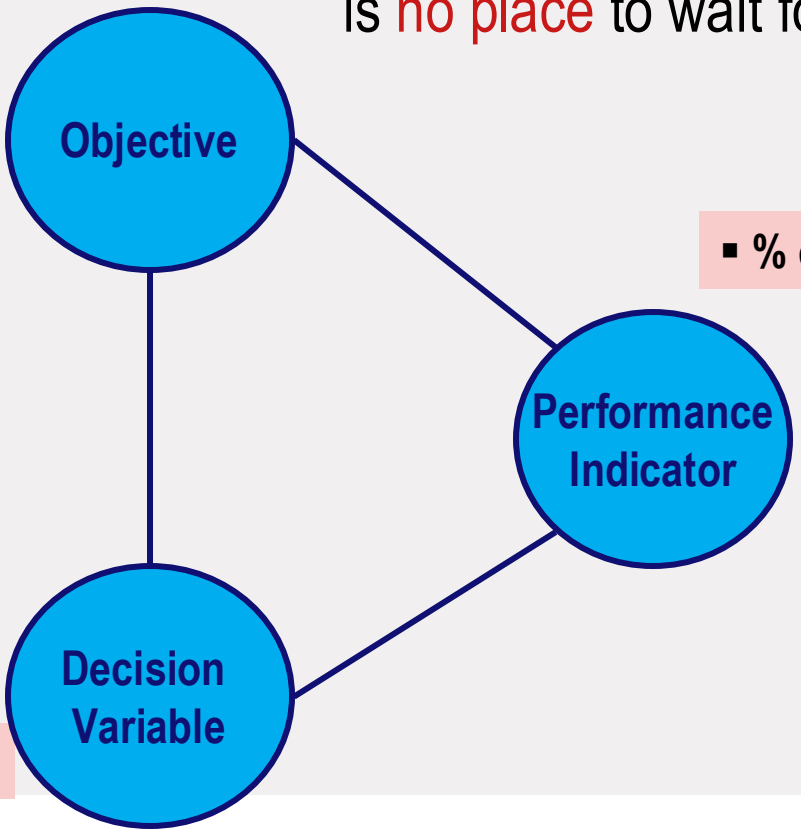


The owner of the petrol station has the feeling that some potential clients are leaving the station because there is no place to wait for service

The Petrol Station – STEP 1 Decision Frame

…the feeling that some potential customers are leaving the station because there is no place to wait for service

- Less customers driving on



- % of customers driving on
- size of queue area

TU/e

The Petrol Station – STEP 2 Black box representation

Environmental Variables

- interarrival time of cars
- service time of cars

Decision variables

- Size of the queue area

PETROL STATION

Output variables

- percentage not served
- (mean waiting time)
- (number not served)

Simulation Methodology (7 steps)

STEP 1: Project definition

STEP 2: Design the simulation study

STEP 3: Conceptual model

STEP 4: Executable model and verification

STEP 5: Validation

STEP 6: Experiments and output analysis

STEP 7: Conclusion

- **model process, objects, and logic**
- **independent of simulation tool used**

Business Process Simulation

Lecture 3a – Conceptual model: main elements

Laura Genga

Department of Industrial Engineering and Innovation Sciences, Information Systems Group

Conceptual model

Define behaviour of simulation model

Tool independent model

Understandable

Unambiguous

Conceptual model

Elements

- Process model
- Information model
- Specifications with logical language
  - Initial state
  - Transition specifications
  - Measurement functions

TU/e

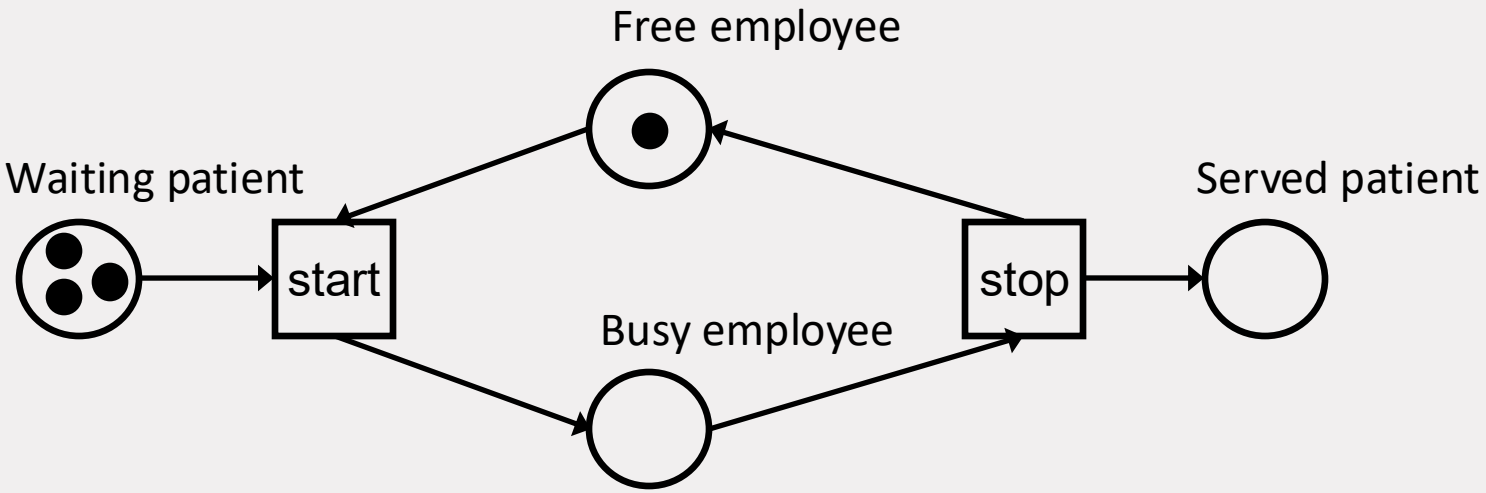
Conceptual model

Process model to specify the behavior

TU/e

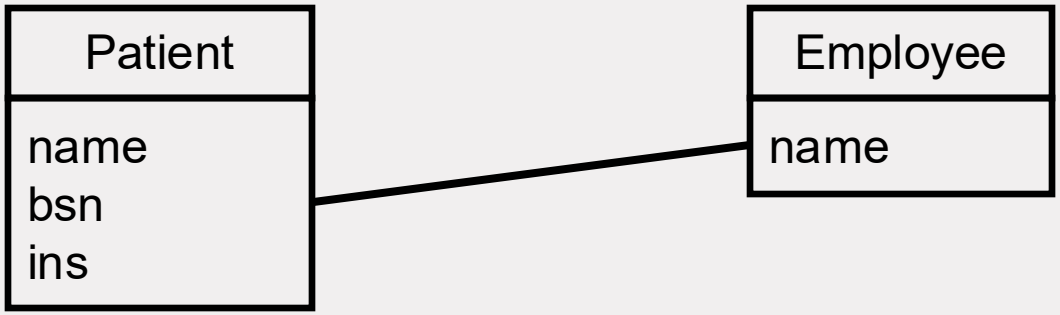
Conceptual model

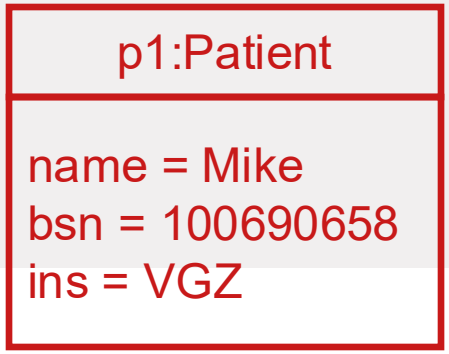
Process model to specify the behavior



Conceptual model

Process model to specify the behavior
Class diagram to specify the information





TU/e

Conceptual model

Process model to specify the behavior
Class diagram to specify the information

Statements in logical language (pseudo code) to specify

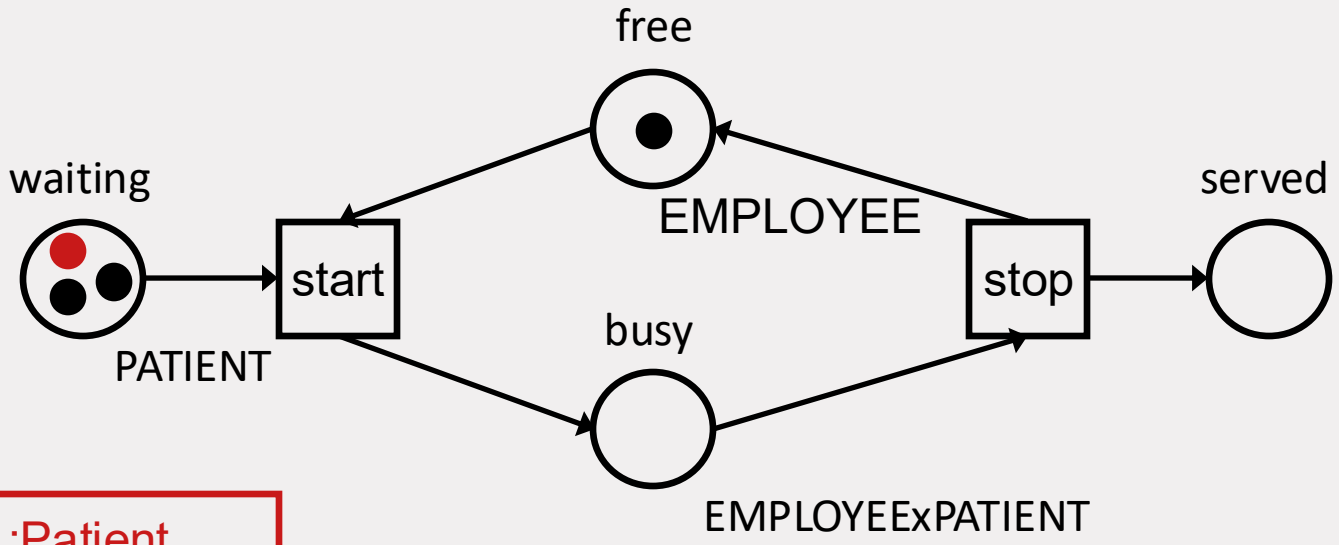
- Initial state of the system
- Transition specifications
- Measurement functions

Logical languages are meant to allow (or enforce) unambiguous statements

TU/e

Conceptual model

Logic to specify: initial state

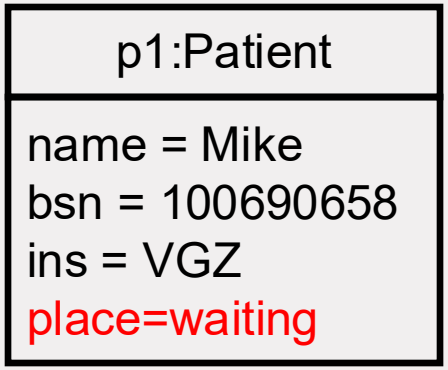


p1:Patient
name = Mike bsn = 100690658 ins = VGZ

```
p1: Patient     p2: Patient     f: Employee
p1.place =      p2.place =      f.place =
waiting         waiting         free
…               …               …
```

Specifying the initial state

Alternative 1

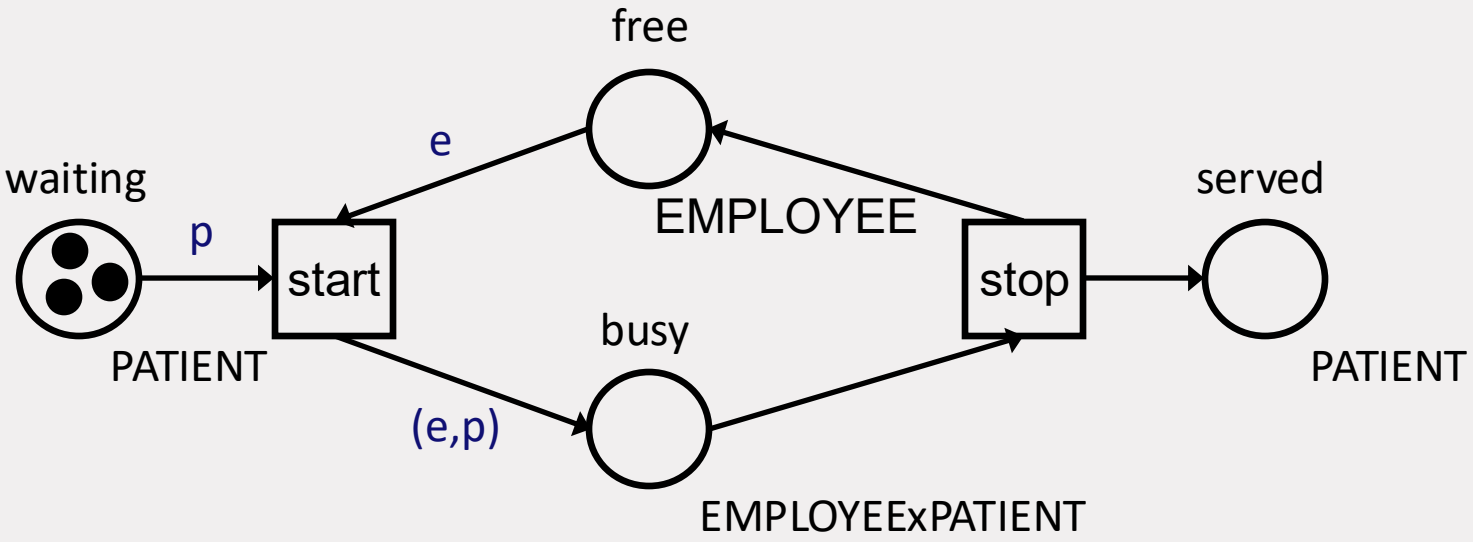


Alternative 2

p1: Patient $\wedge$
p1.name = Mike $\wedge$
p1.bsn = 100690658 $\wedge$
p1.ins = VGZ $\wedge$
p1.place = waiting

Conceptual model

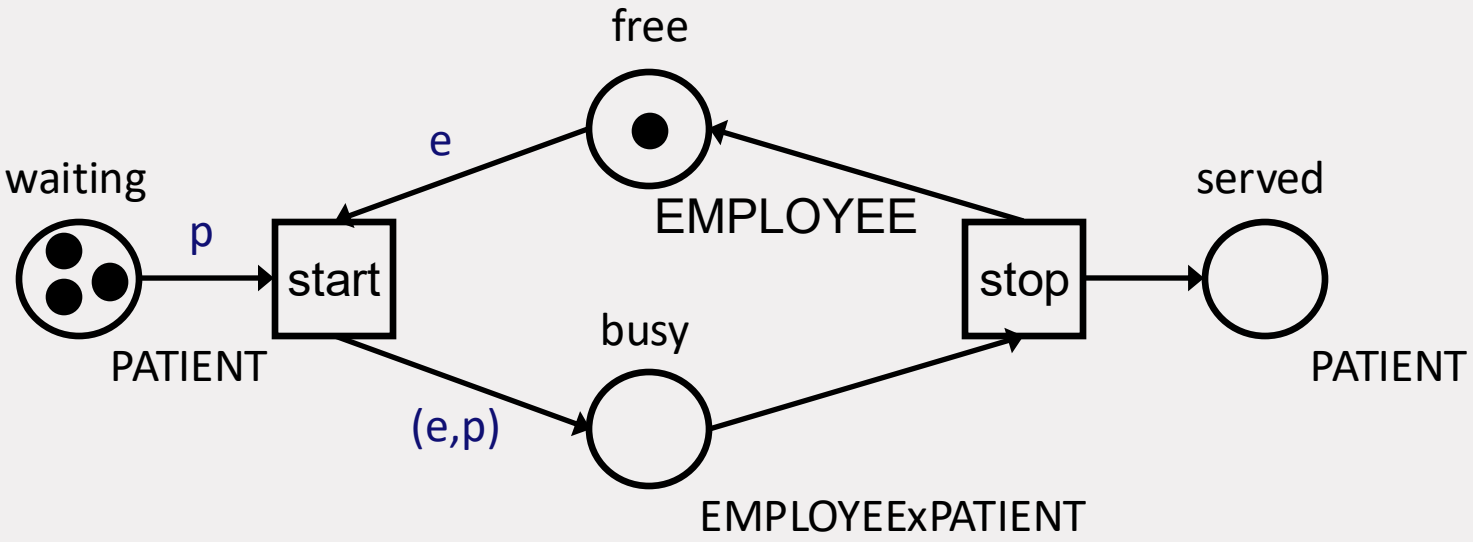
Logic to specify operations (transitions)




```
Start
Pre:    p in waiting
Post:   waiting :=     waiting.remove(p)
        free    :=     free.remove(e)
        busy    :=     busy.add(e,p)
```

Conceptual model

Logic to specify measurement functions

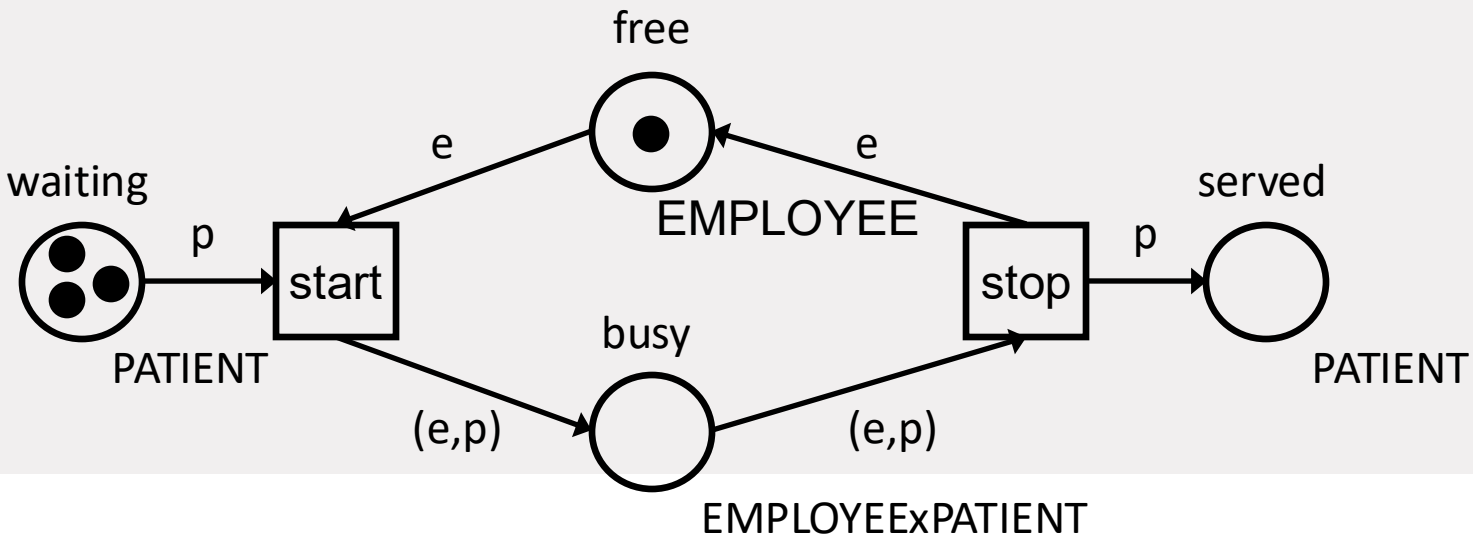


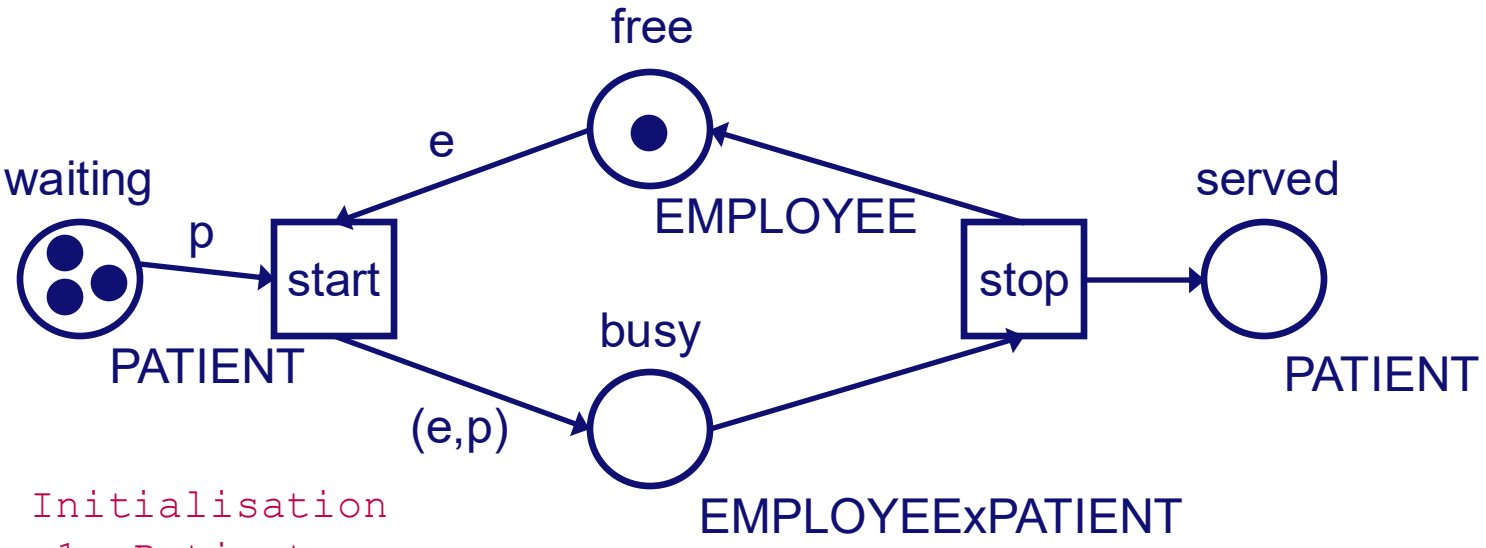

```
NrServed        =  served.Length;
AverageTPT      =  Sum(p.throughputtime: p in Served) / NrServed
```

Conceptual model

Process model to specify the behavior
Class diagram to specify the information
Logic to specify:

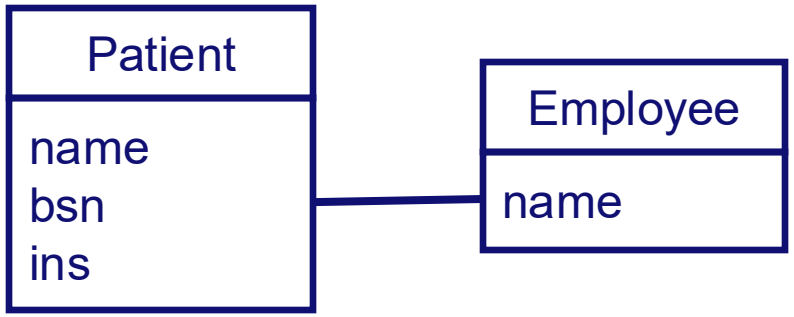
- Initial state
- Operations
- Measurement functions






```
Initialisation
p1: Patient ∧
p1.name = Mike ∧
p1.bsn = 100690658 ∧
p1.ins = VGZ ∧
p1.place = waiting

f: Employee
f.Place = free
```



…

```
Start
Pre:    p in waiting
Post:   waiting waiting.remove(p)
        free    :=free.remove(e)
        busy    :=busy.add(e,p)
```

```
NrServed     =  served.Length;
AverageTPT   =  Sum(p.throughputtime: p in Served) /
                NrServed
```

p1:Patient
name = Mike bsn = 100690658 ins = VGZ place = waiting

Business Process Simulation

Lecture 3b – Simple queuing system : process and information model

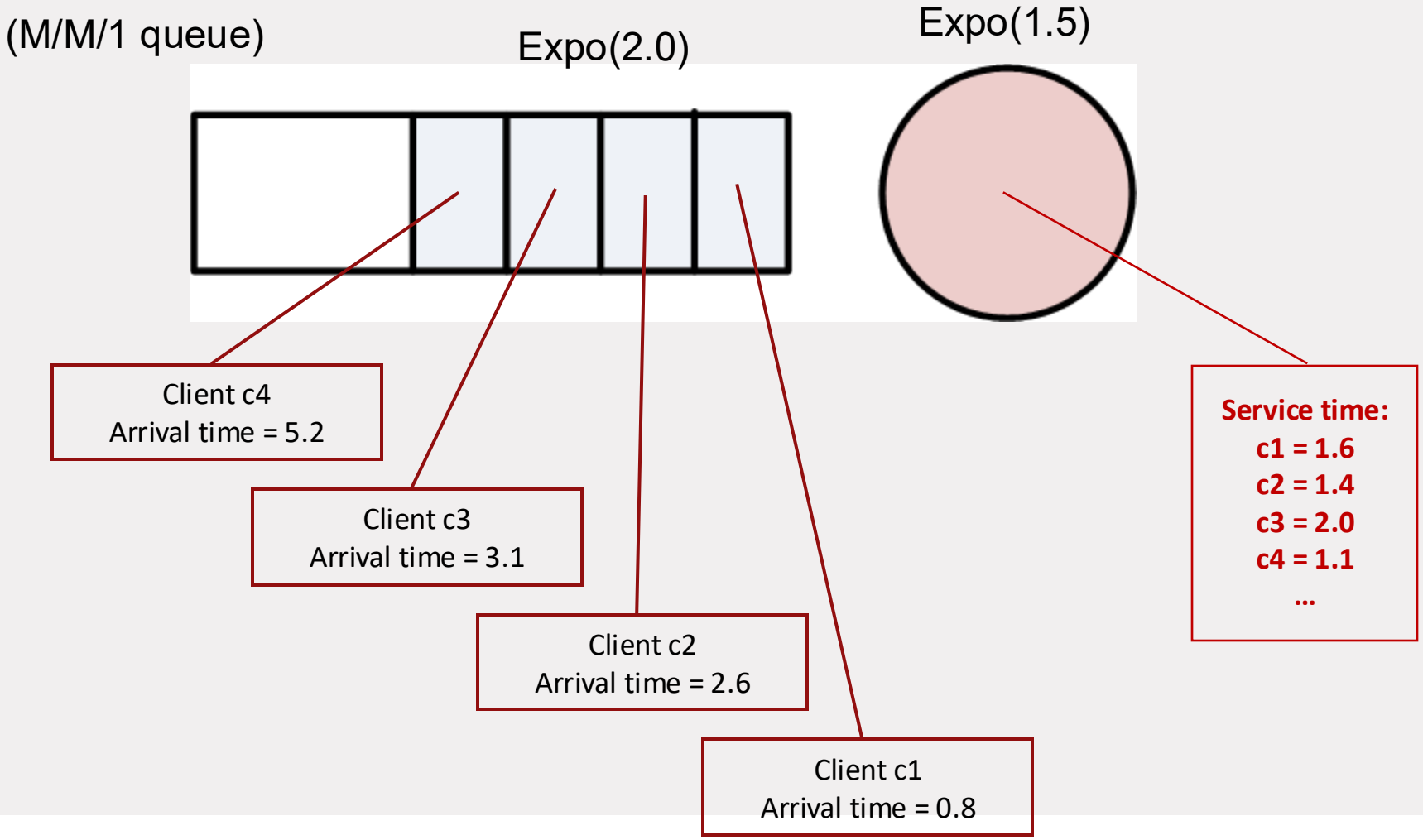
Laura Genga

Department of Industrial Engineering and Innovation Sciences, Information Systems Group

Overview on lecture modules

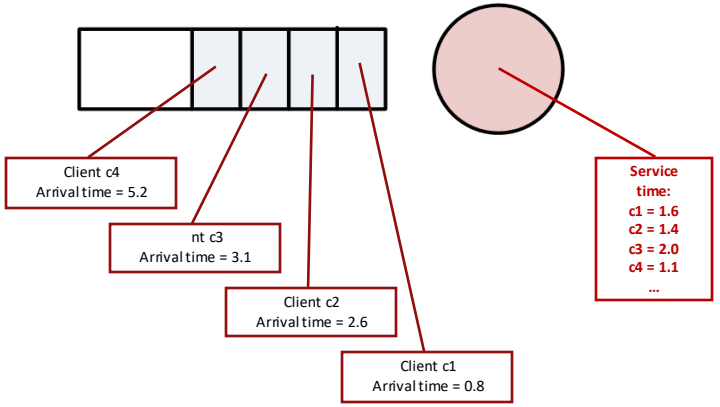
a) Conceptual model: main elements
b) Simple queuing system: process and information model
c) Simple queuing system: transition specifications
d) How does simulation work?
e) Petrol station example and manual simulation
f) Additional examples

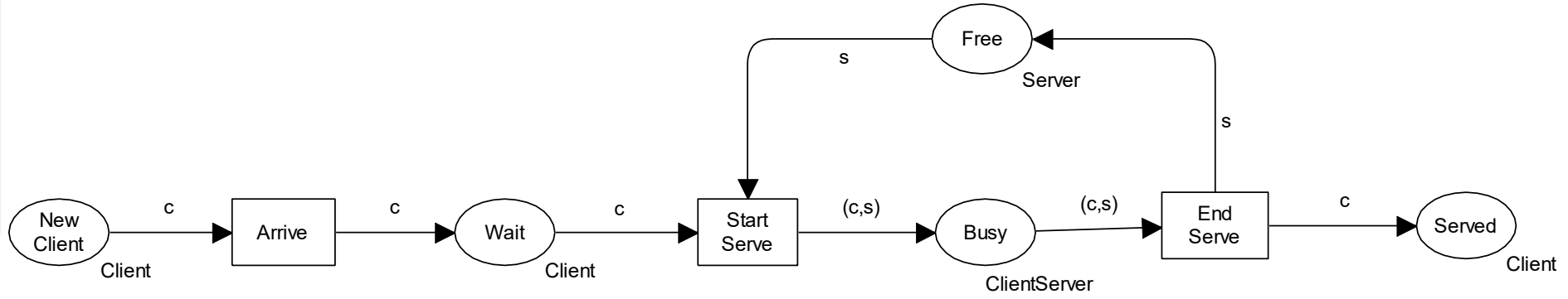
Simple queueing system



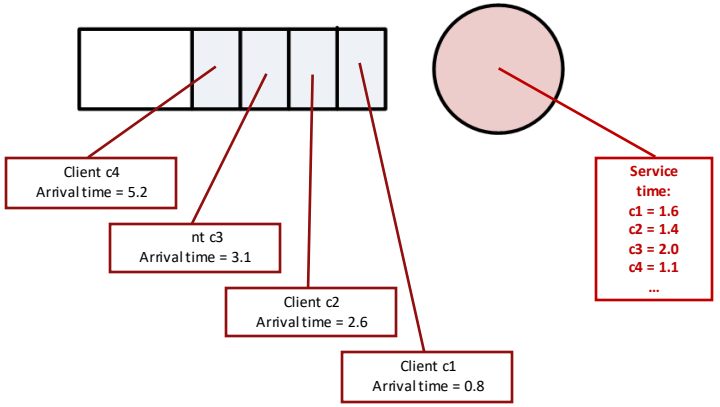
Simple queueing system

Specify behavior



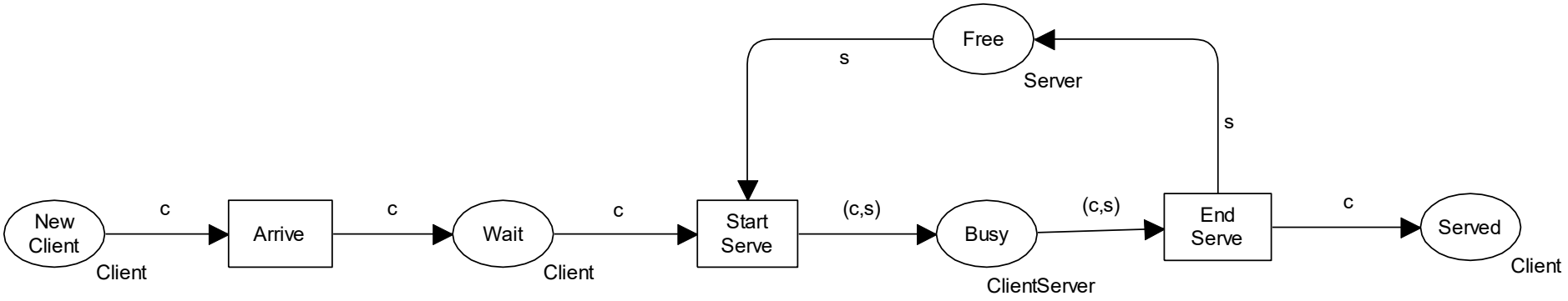


Simple queueing system

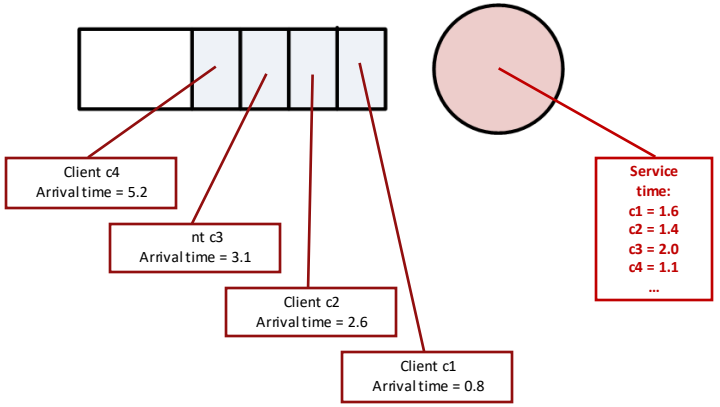


Classical Petri Net:

- Transition fires if token available in each input place
- Tokens are objects
- Type of token specified by type of place
- Arc labels indicate flow of tokens

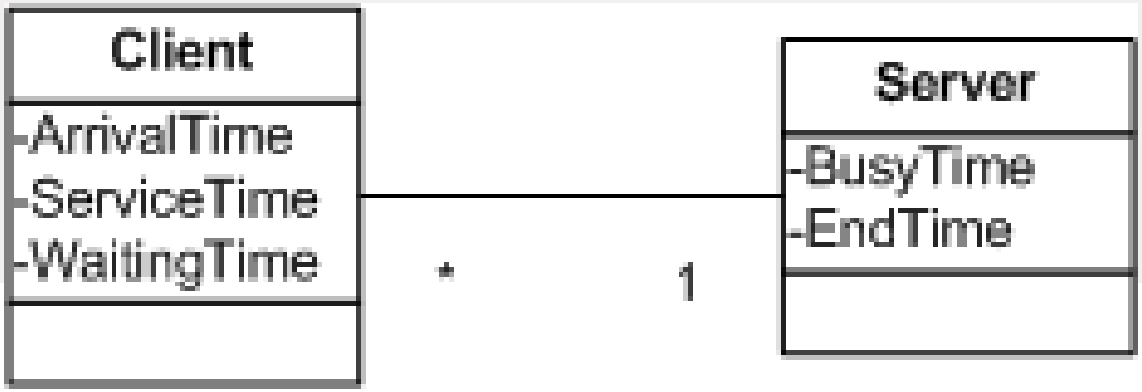


Simple queueing system

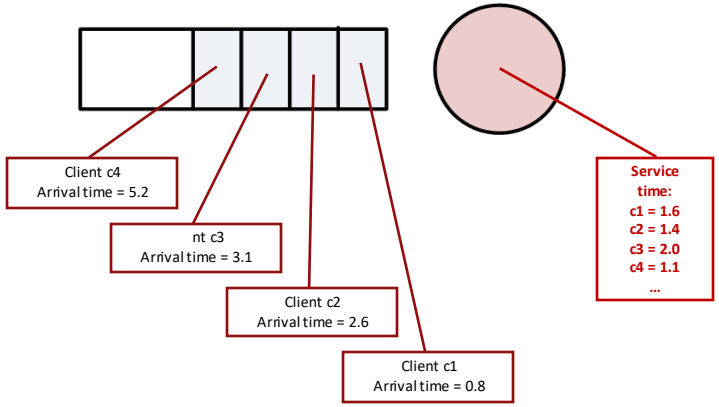


Specify information

- Object classes
- Attributes
- Associations



Simple queueing system



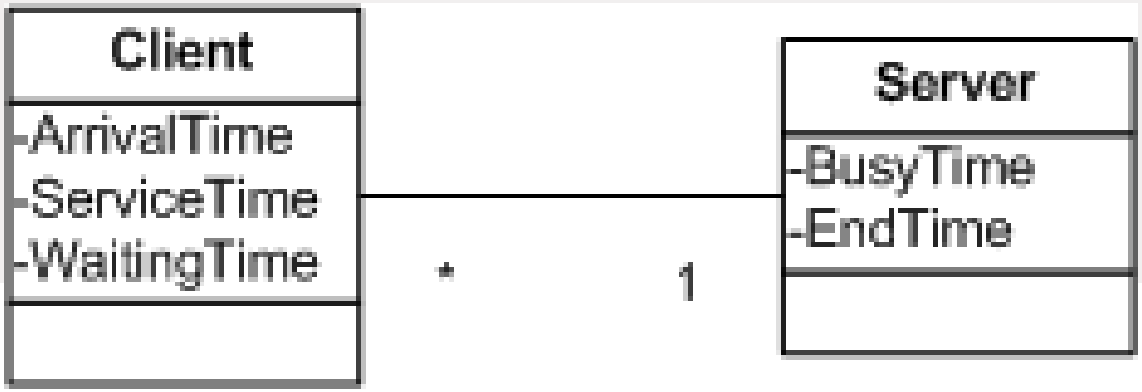
c1 : Client
ArrivalTime = 0.8
ServiceTime = 1.6
WaitingTime = 0.0

c2 : Client
ArrivalTime = 2.6
ServiceTime = 1.4
WaitingTime = 0.0

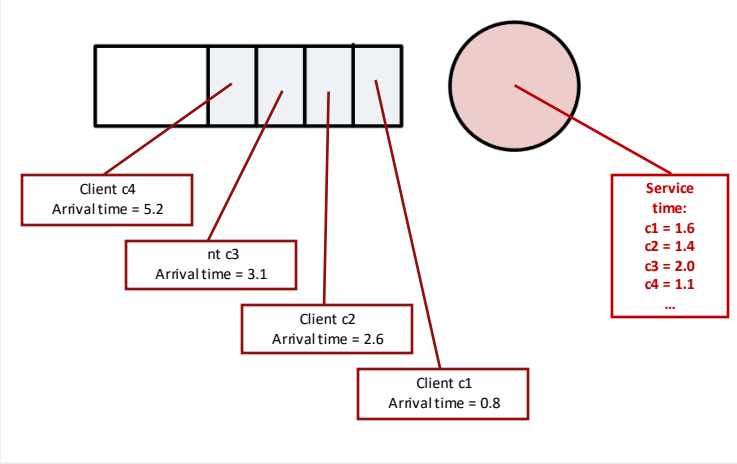
c3 : Client
ArrivalTime = 3.1
ServiceTime = 2.0
WaitingTime = 0.0

c4 : Client
ArrivalTime = 5.2
ServiceTime = 1.1
WaitingTime = 0.0

s1 : Server
BusyTime = 0.0
EndTime = 0.0



Simple queueing system



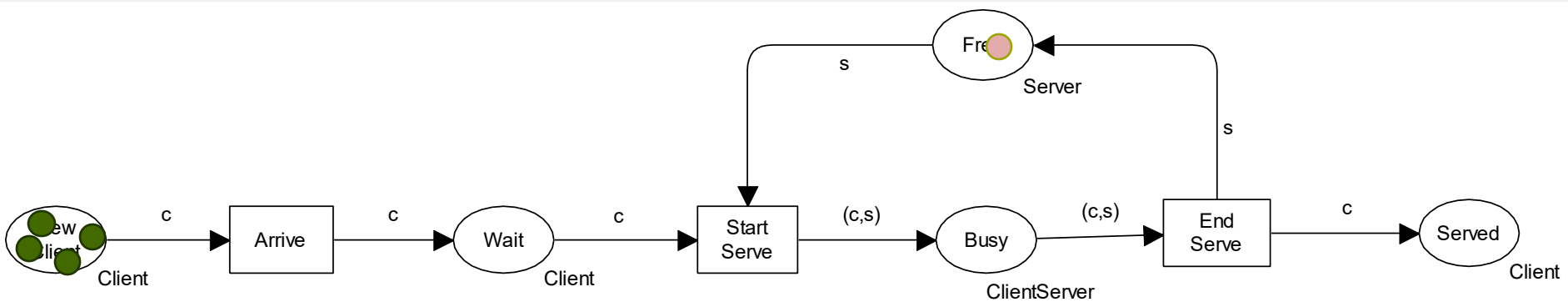
c1 : Client
ArrivalTime = 0.8
ServiceTime = 1.6
WaitingTime = 0.0

c2 : Client
ArrivalTime = 2.6
ServiceTime = 1.4
WaitingTime = 0.0

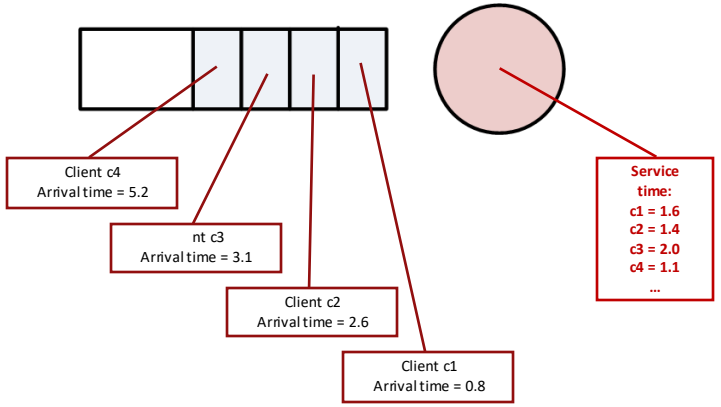
c3 : Client
ArrivalTime = 3.1
ServiceTime = 2.0
WaitingTime = 0.0

c4 : Client
ArrivalTime = 5.2
ServiceTime = 1.1
WaitingTime = 0.0

s1 : Server
BusyTime = 0.0
EndTime = 0.0



Simple queueing system



Some remarks on notation:

Creating objects

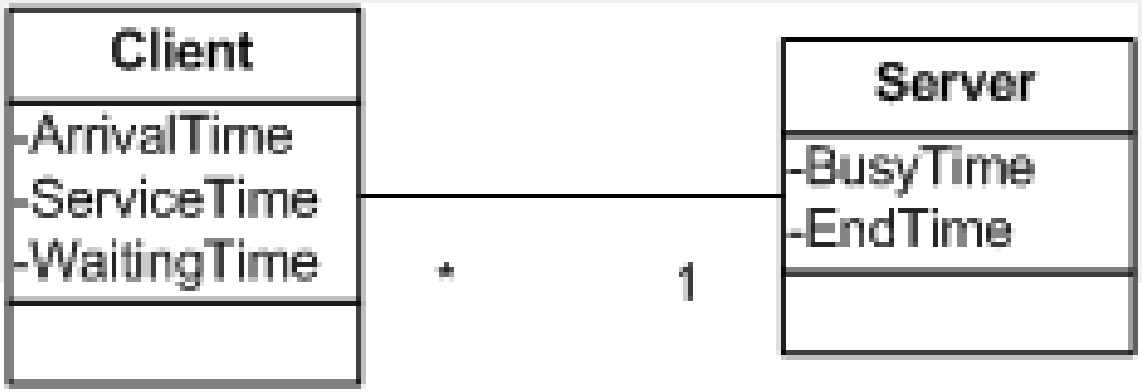
c1: Client $\wedge$ c2: Client $\wedge$ c3: Client $\wedge$ c4: Client

s1: Server

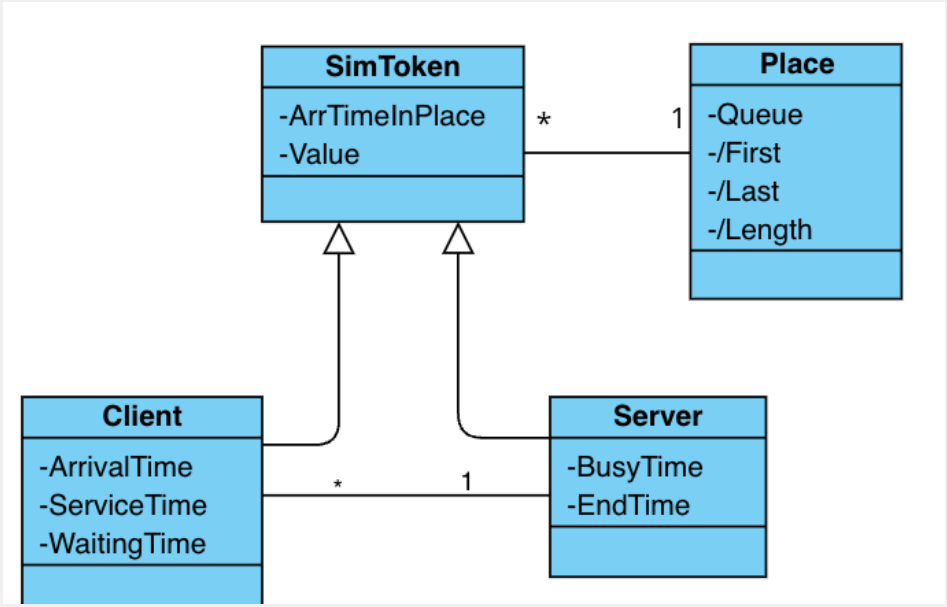
Creating links

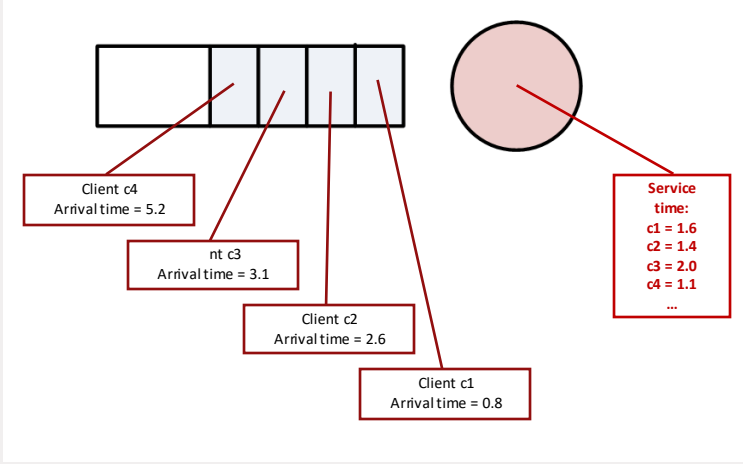
c1.Server = s1

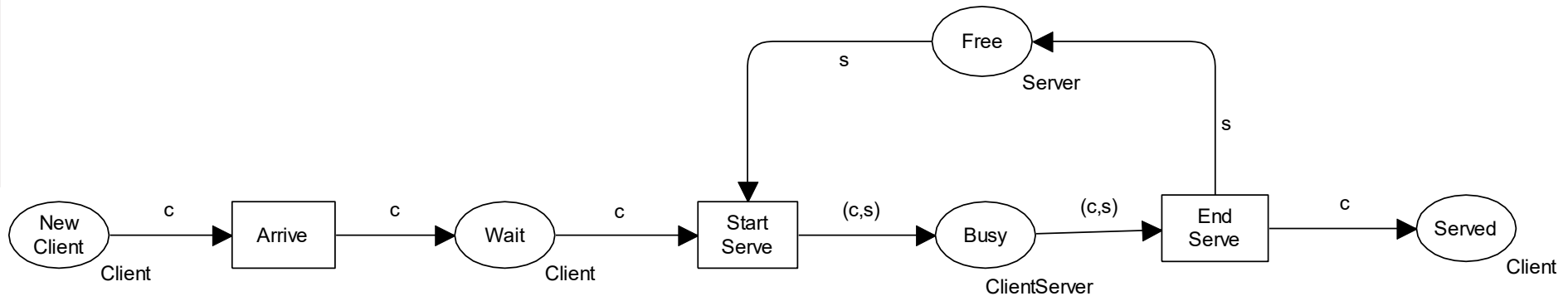
c1 in s1.Client



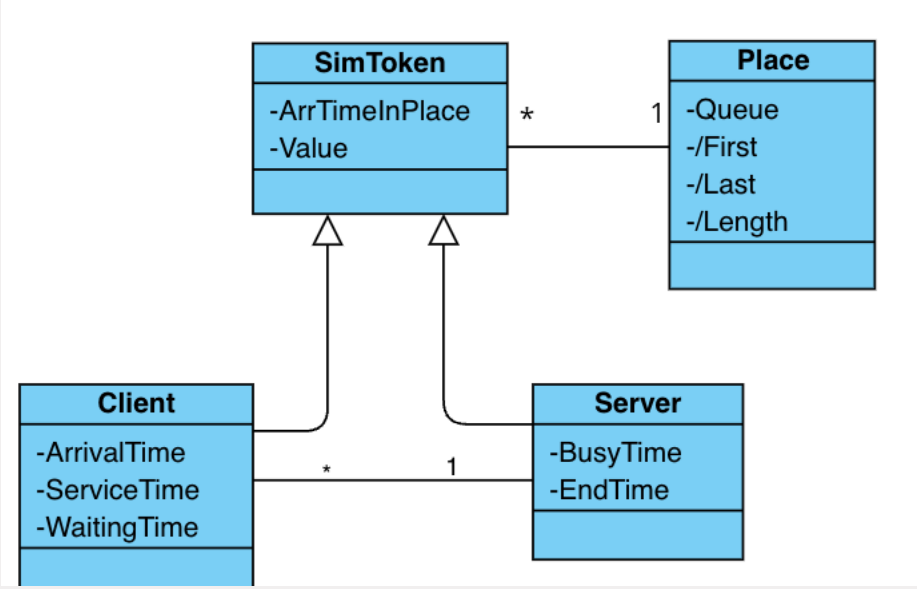
Simple queueing system

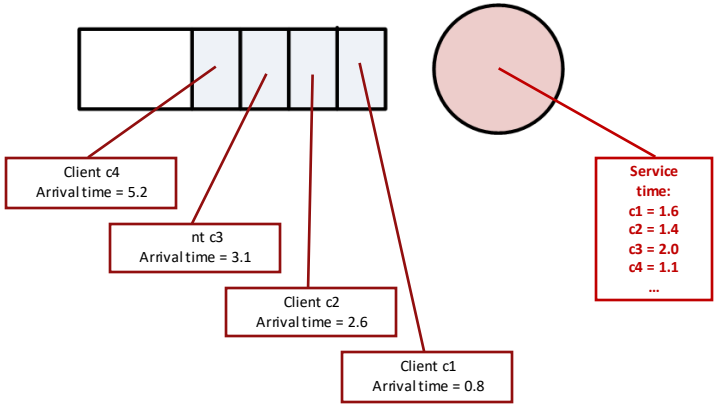






Simple queueing system





Constraints

Place

self.Length = len(self.Queue) ∧

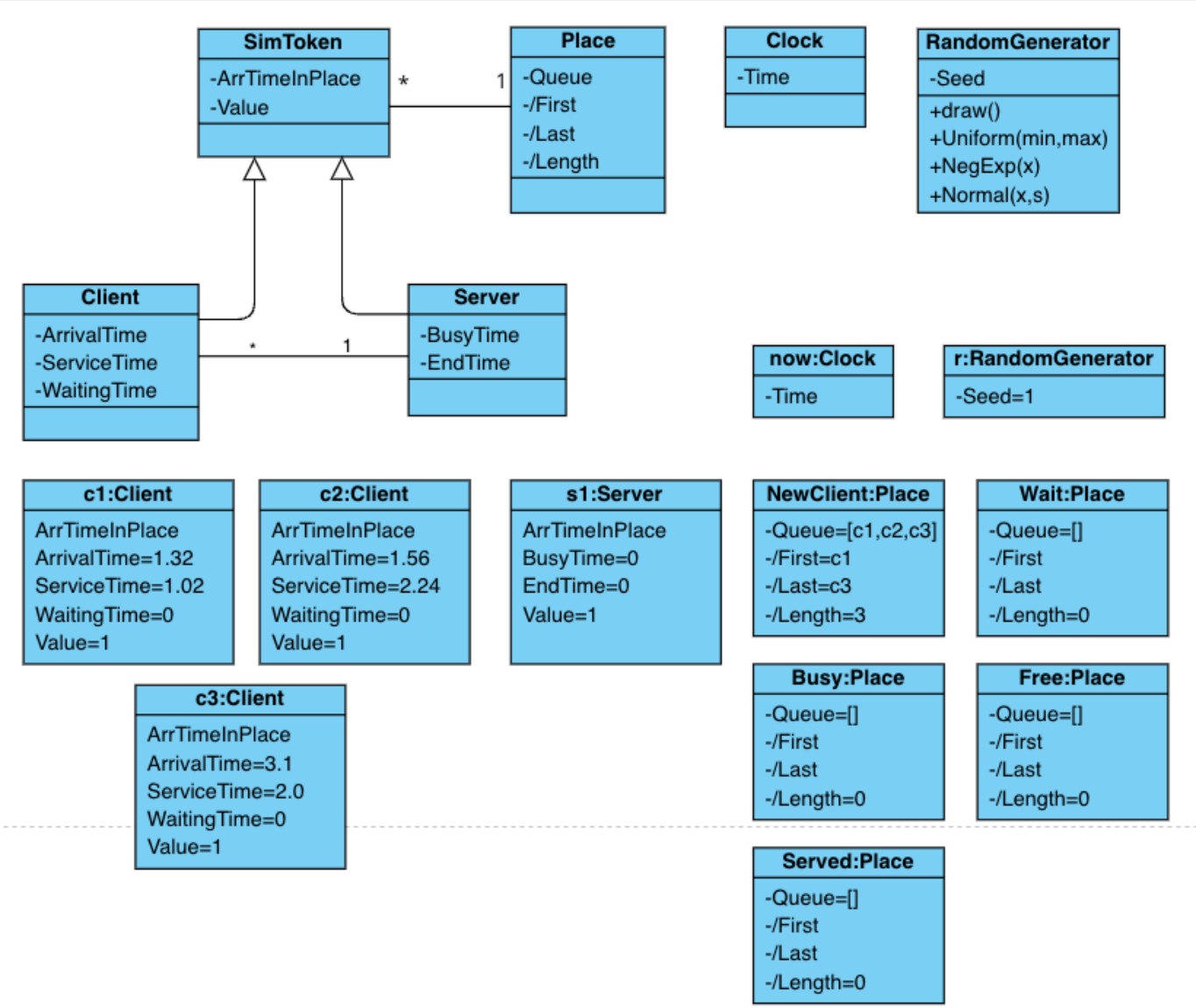
self.First = {o in self.Queue:

o.ArrTimeInPlace = Min { t.ArrTimeInPlace: t in self.Queue } } ∧

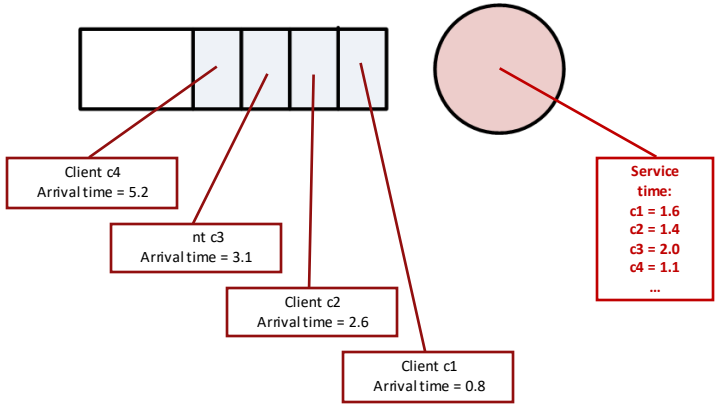
self.Last = {o in self.Token:

o.ArrTimeInPlace = Max { t.ArrTimeInPlace: t in self.Queue } } .

Simple queueing system



Simple queueing system

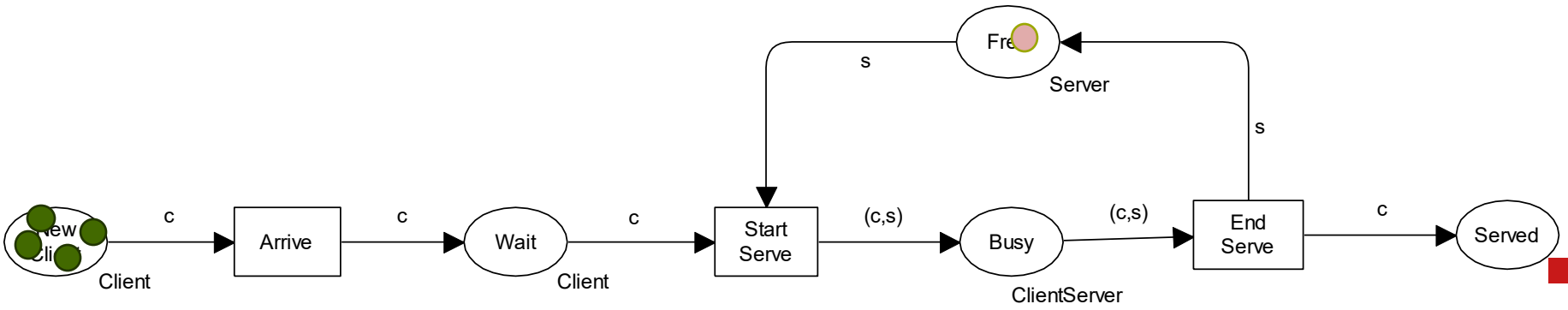


Initial state

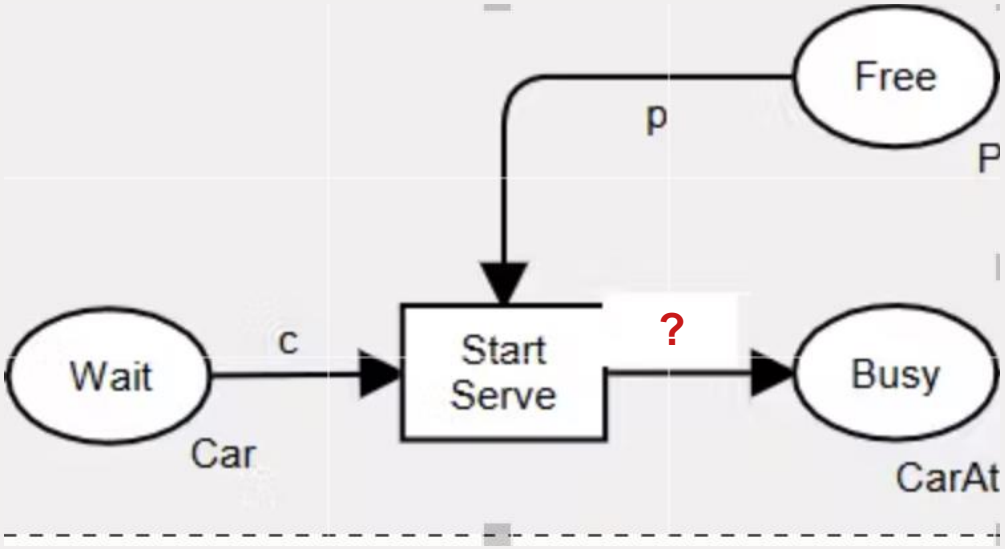
c1: Client ∧
c1.Place = NewClient ∧
c1.ArrivalTime = 0.8 ∧
c1.ServiceTime = 1.6 ∧
c1.WaitingTime = 0.0 ∧
c2: Client ∧
…

s1: Server ∧
s1.Place = Free ∧
s1.BusyTime = 0.0 ∧

now: Clock ∧ % now is an instance of a Clock object %
now.Time = 0 ∧
r: RandomGenerator ∧ % r is an instance of a random number generator %
r.Seed = 1; % every block is terminated with a ; %



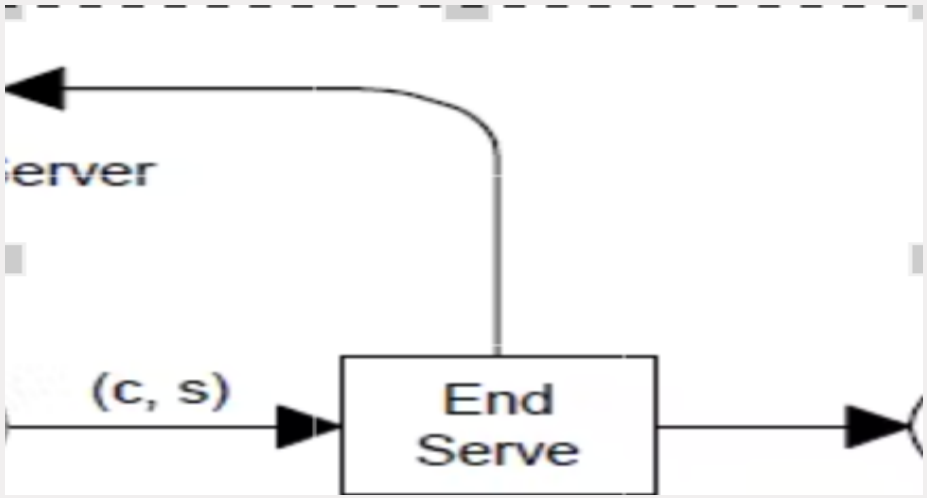
Which token(s) do we have on the output edge of "start serve"



www.menti.com
Code: **21673354**

TU/e

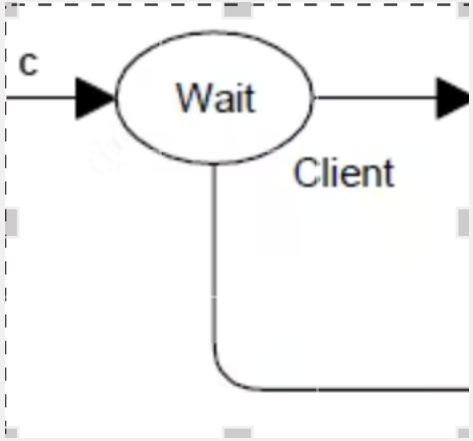
Which token(s) do we have on each outgoing edge?



www.menti.com
Code: **21673354**

TU/e

Which token(s) do we have on each outgoing edge?



www.menti.com
Code: **21673354**

TU/e

1BK20 Business Process Simulation

1BK20 Lecture 3c – Simple queuing system : transition specifications

Laura Genga

Department of Industrial Engineering and Innovation Sciences, Information Systems Group

Overview on lecture modules

a) Conceptual model: main elements
b) Simple queuing system: process and information model
c) Simple queuing system: transition specifications
d) How does simulation work?
e) Petrol station example and manual simulation
f) Additional examples

Transition specifications

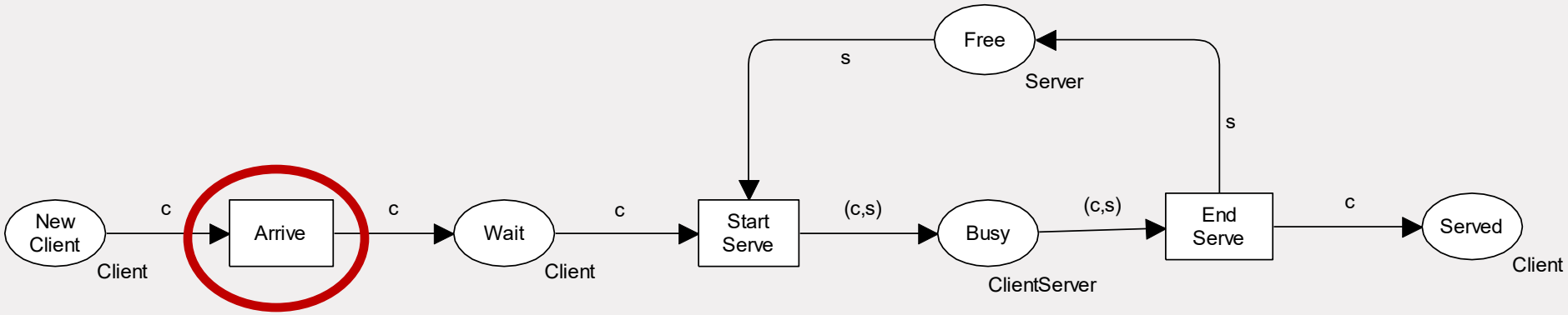
Pre conditions

- Describes when a transition may fire
- Reference to any object is allowed

Post conditions

- Describes the state of the system after the transition has fired

Simple queueing system – specifications



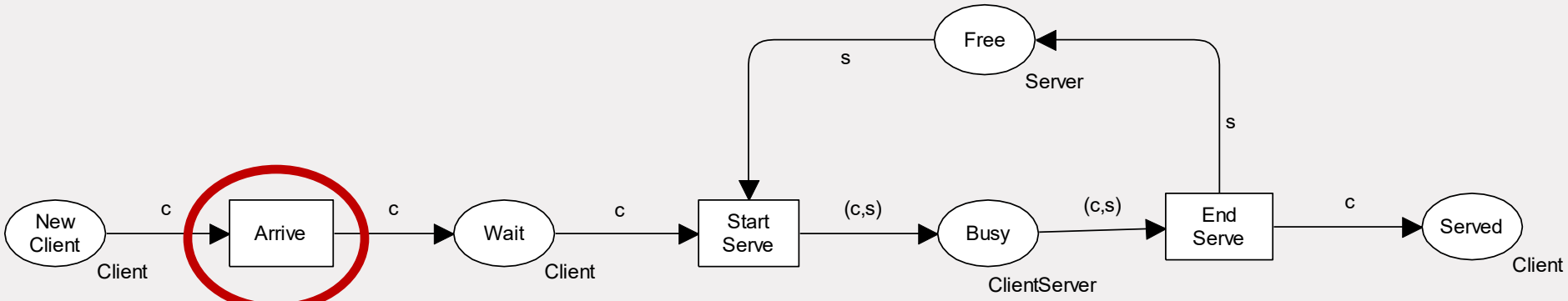
Arrive

Pre: c in New Client $\wedge$ c.ArrivalTime = now.Time

Post:NewClient := NewClient.remove(c)

Wait := Wait.add(c)

Simple queueing system – specifications



Not necessary to be mentioned all the time

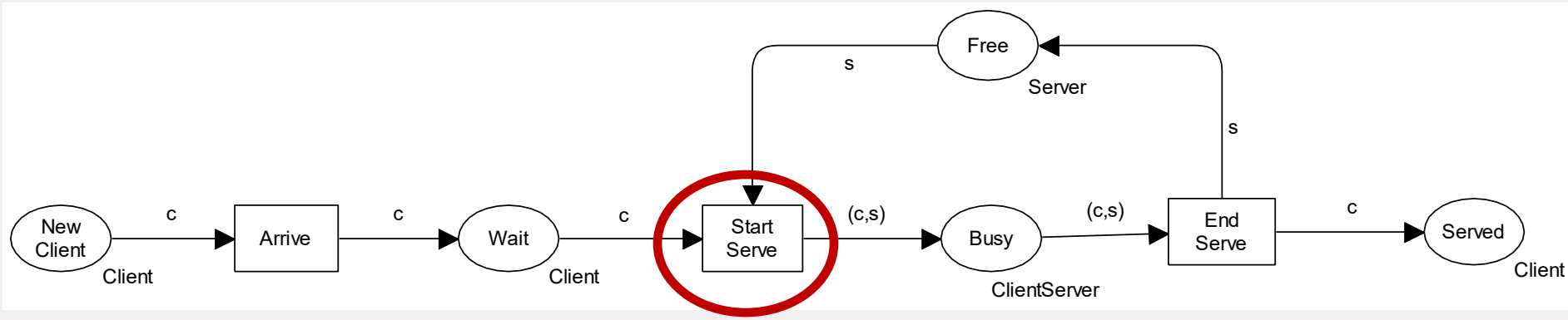
Arrive

Pre: ~~(c in New Client)~~ ∧ c.ArrivalTime = now.Time

Post: ~~NewClient := NewClient.remove(c)~~ ∧

Wait := Wait.add(c)

Simple queueing system – specifications



StartServe

Pre ~~(c in Wait ∧ s in Free ∧)~~ c in Wait.First

Post ~~(Wait := Wait.remove(c) ∧ Free := Free.remove(s) ∧~~

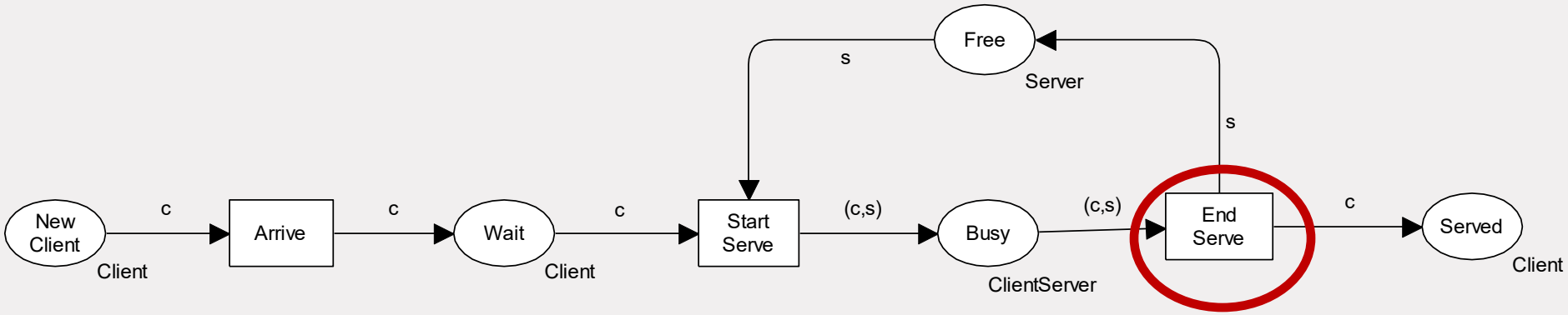
Busy.add(c,s)

c.WaitingTime := now.Time - c.ArrivalTime ∧

s.EndTime := now.Time + c.ServiceTime ∧

s.BusyTime := s.BusyTime + c.ServiceTime;

Simple queueing system – specifications



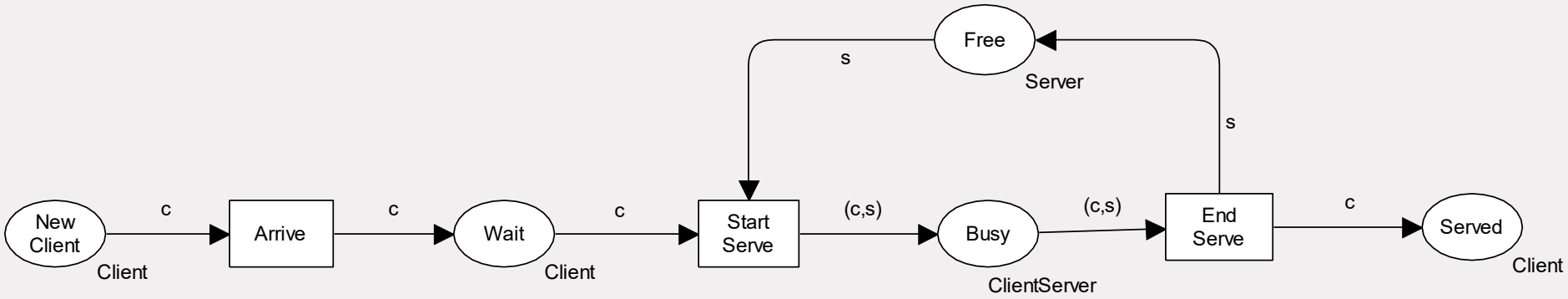
EndServe

Pre ~~(c,s) in Busy ∧~~ now.Time = s.EndTime

Post Served :=Served.add(c) ∧

Free := Free.add(s);

Simple queueing system – specifications



Functions

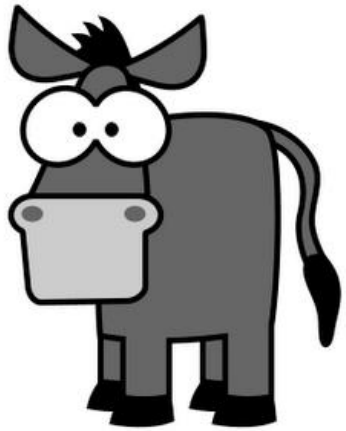
NrServed	=	Served.Length;
AverageWait	=	Sum(c.Waitingtime: c in Served.Queue) / NrServed;
Occupation	=	(s1.BusyTime / now.Time) * 100%;

Pre- and post-condition language

Some remarks:

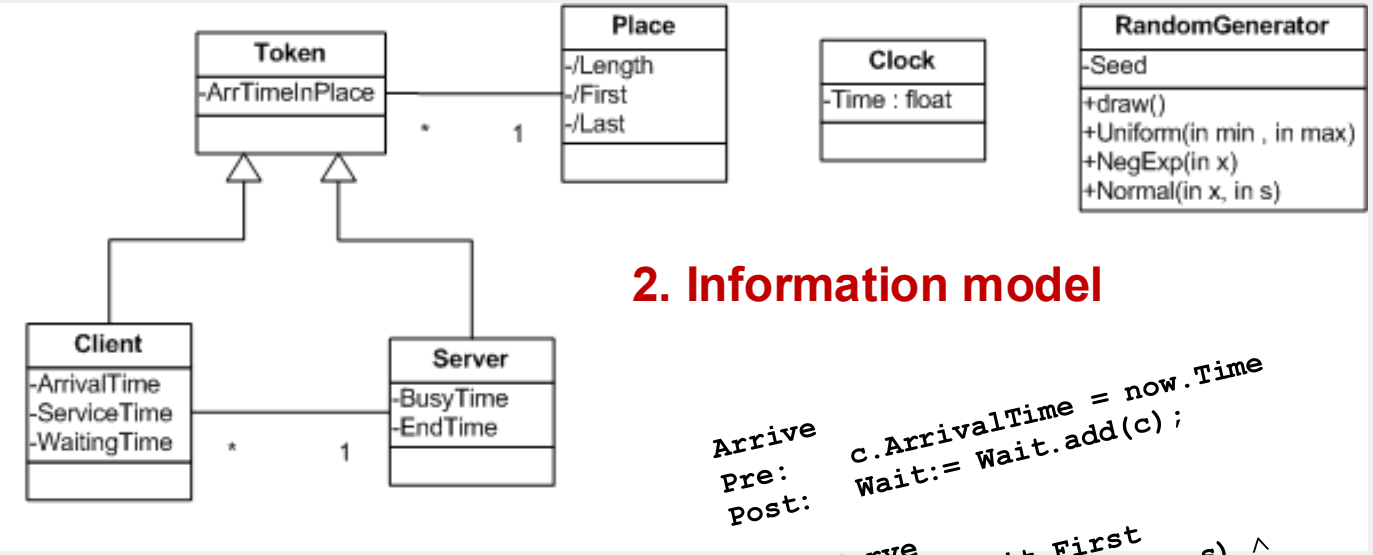
- Any variable that is not mentioned as left operand in a := operation, remains unchanged;
- Random number generator will generate a new random number every time it is evaluated in a post-condition
- Random number generator can not be evaluated in a pre-condition!

Specification language



Need for a formal (logical) language for specifications to prevent ambiguity in declarations misunderstandings

Conceptual model



2. Information model

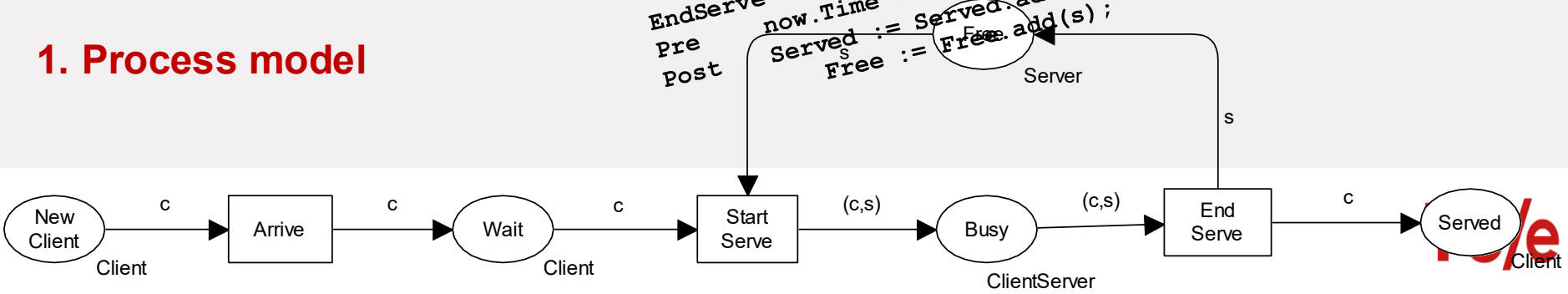
```
Arrive
Pre:    c.ArrivalTime = now.Time
Post:   Wait:= Wait.add(c);

StartServe
Pre     c in Wait.First
Post    Busy:= Busy.add(c,s) ^
        c.WaitingTime := now.Time - c.ArrivalTime ^
        s.EndTime := now.Time + c.ServiceTime ^
        s.BusyTime := s.BusyTime + c.ServiceTime;

EndServe
Pre     now.Time = s.EndTime
Post    Served := Served.add(c)
        Free := Free.add(s);
```

3. Specifications

1. Process model



1BK20 Business Process Simulation

1BK20 Lecture 3d – How does simulation work?

Laura Genga

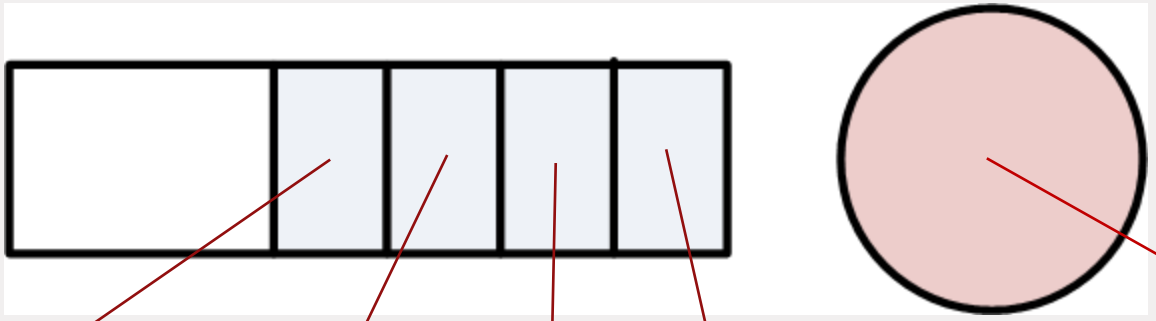
Department of Industrial Engineering and Innovation Sciences, Information Systems Group

Overview on lecture modules

a) Conceptual model: main elements
b) Simple queuing system: process and information model
c) Simple queuing system: transition specifications
d) How does simulation work?
e) Petrol station example and manual simulation
f) Additional examples

How does simulation work?

(M/M/1 queue)



Client c4
Arrival time = 5.2

Client c3
Arrival time = 3.1

Client c2
Arrival time = 2.6

Client c1
Arrival time = 0.8

Service time:
c1 = 1.6
c2 = 1.4
c3 = 2.0
c4 = 1.1
...

c1 : Client
ArrivalTime = 0.8
ServiceTime = 1.6
WaitingTime = 0.0

c2 : Client
ArrivalTime = 2.6
ServiceTime = 1.4
WaitingTime = 0.0

c3 : Client
ArrivalTime = 3.1
ServiceTime = 2.0
WaitingTime = 0.0

c4 : Client
ArrivalTime = 5.2
ServiceTime = 1.1
WaitingTime = 0.0

s1 : Server
BusyTime = 0.0
EndTime = 0.0

TU/e

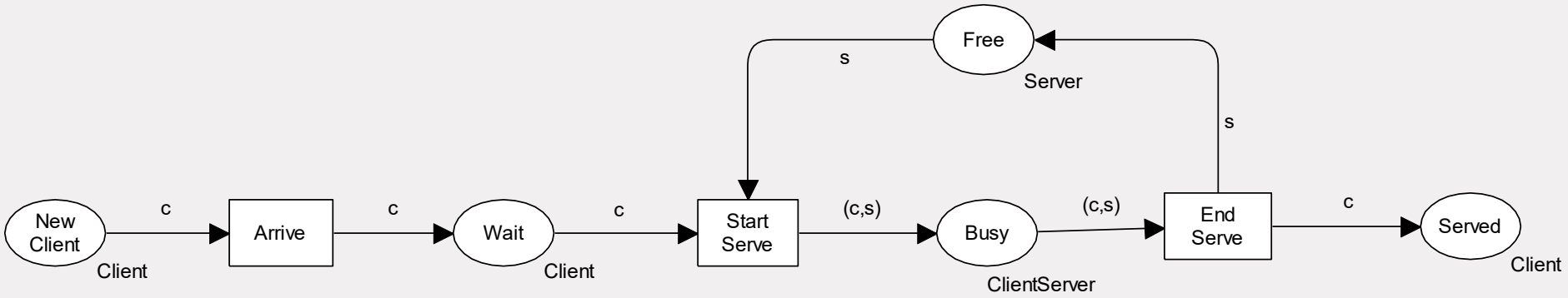
How does simulation work?

Simulation clock

Next event technique:

- select and perform the first next event
- move the simulation clock to the time of that event
- compute further next events and put them in the queue

Events and transitions



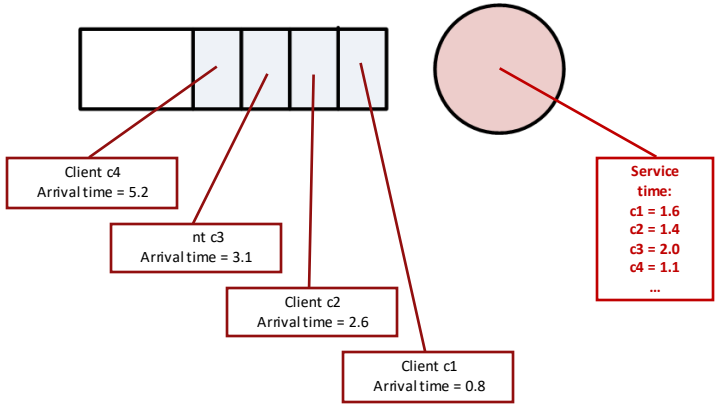
Arrive
Pre: c.ArrivalTime = now.Time
Post:…

StartServe
Pre c in Wait.First
Post ….

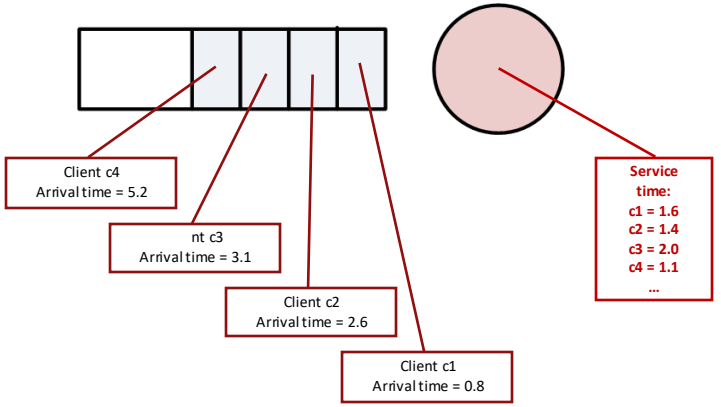
EndServe
Pre now.Time := s.EndTime
Post ….

Time line

Time line describes the events and their order for the simple queueing system



Time line

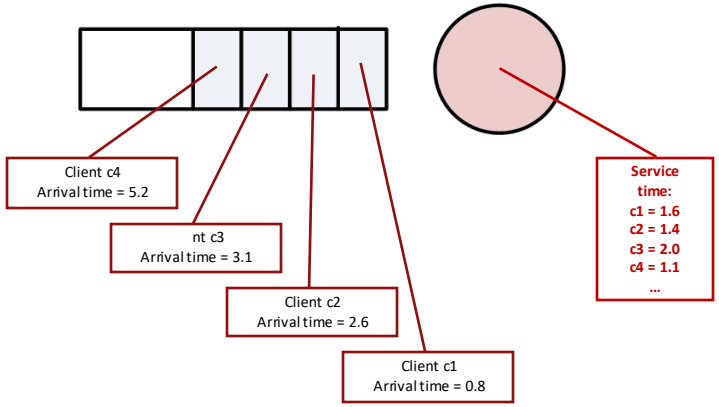


- **Next arrival event:**
  - **c1 at time 0.8**
  - **c2 at time 2.6**
  - **c3 at time 3.1**
  - **c4 at time 5.2**

- **Next server event:**
  - -

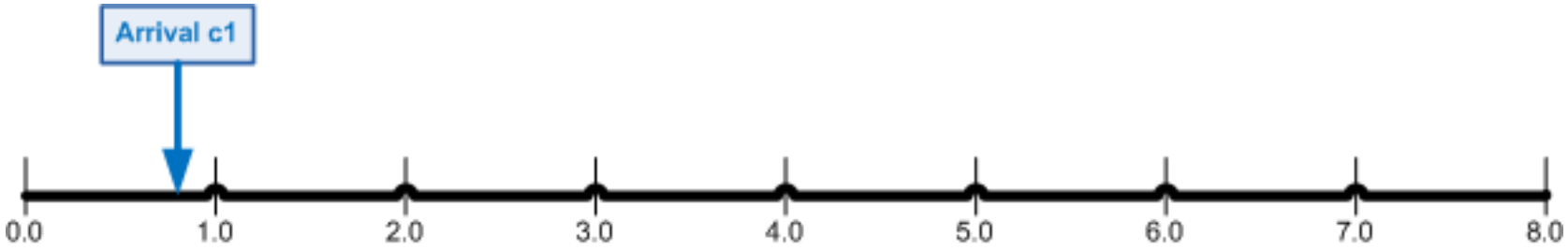
time = 0.0

Time line



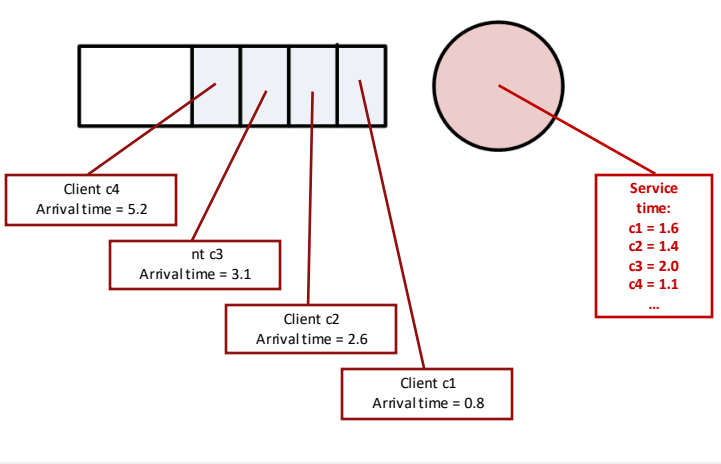
- **Next arrival event:**
  - **c2 at time 2.6**
  - **c3 at time 3.1**
  - **c4 at time 5.2**

- **Next server event:**
  - **Start c1 at time 0.8**



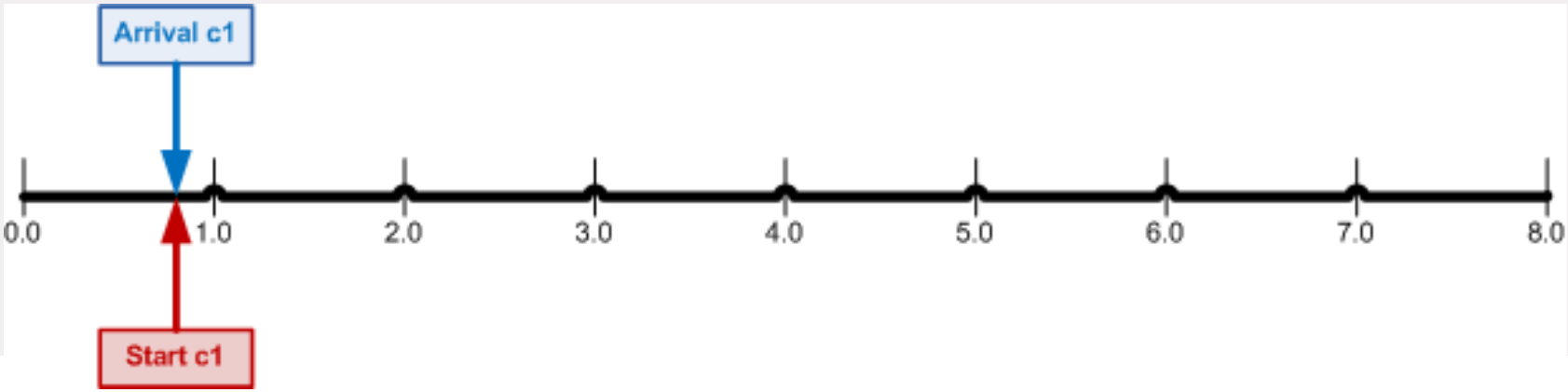
time = 0.8

Time line



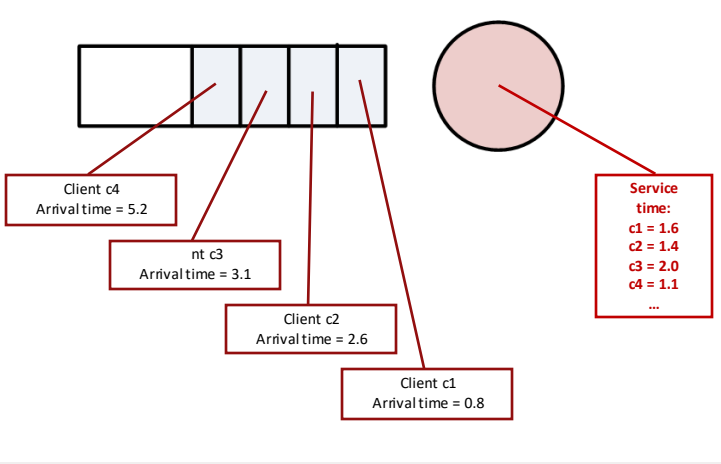
- **Next arrival event:**
  - **c2 at time 2.6**
  - **c3 at time 3.1**
  - **c4 at time 5.2**

- **Next server event:**
  - **end c1 at time 2.4 (= 0.8 + 1.6)**



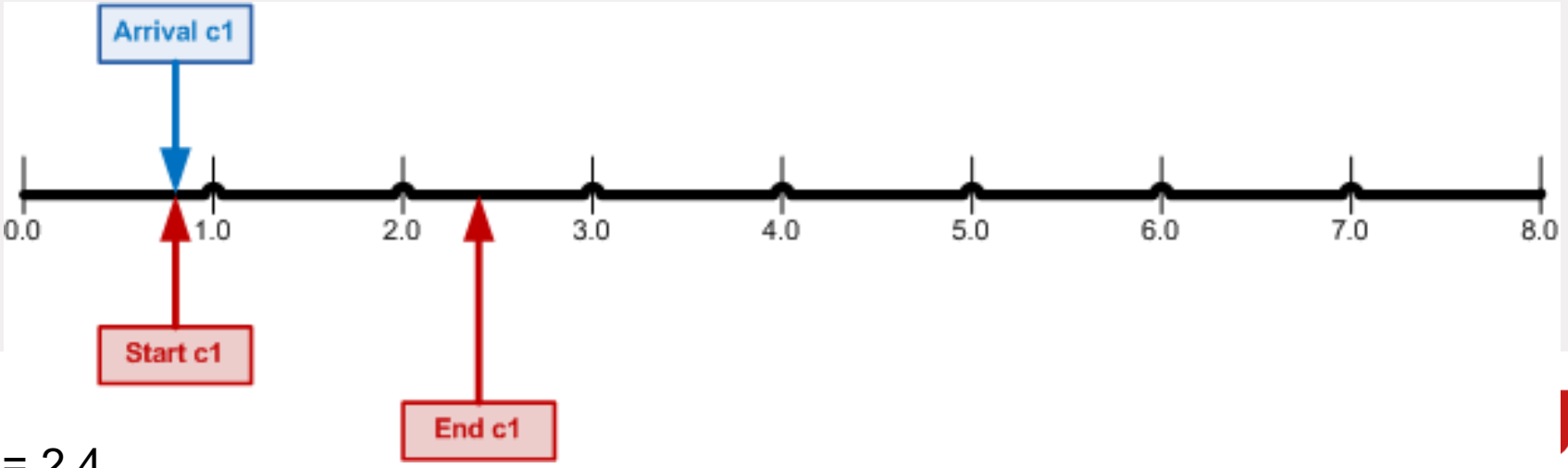
time = 0.8

Time line



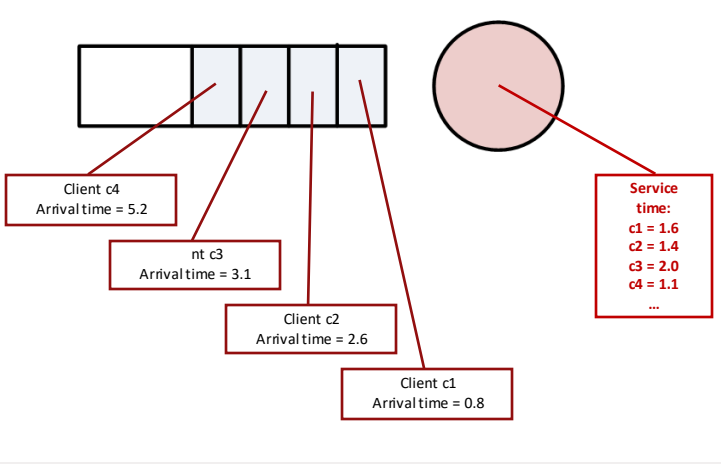
- **Next arrival event:**
  - **c2 at time 2.6**
  - **c3 at time 3.1**
  - **c4 at time 5.2**

- **Next server event:**
  - **-**



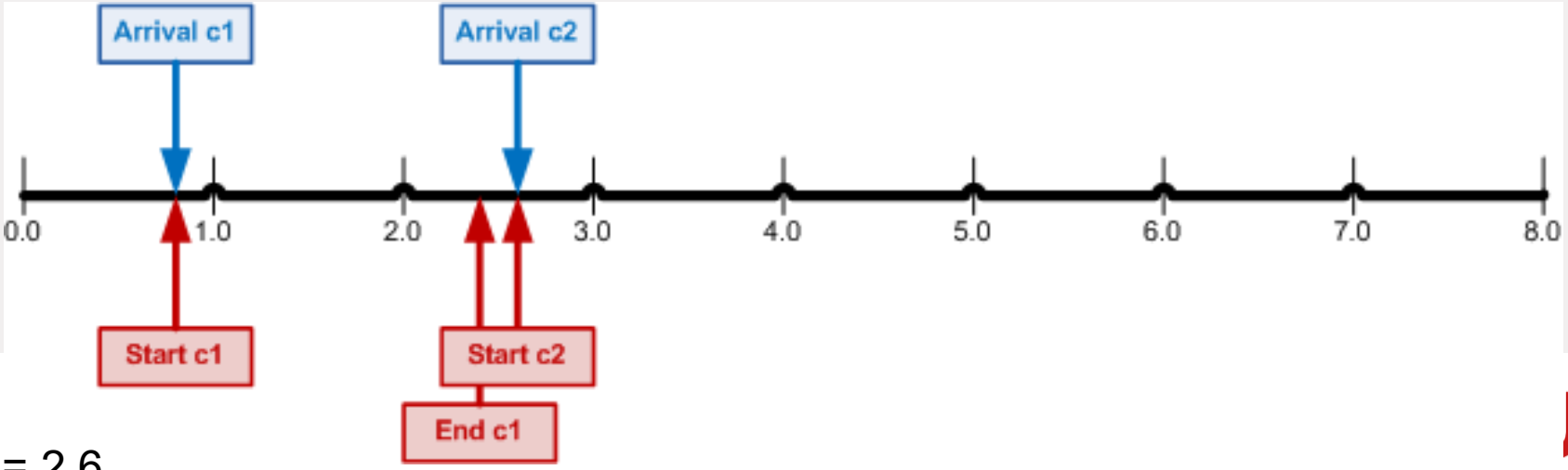
time = 2.4

Time line



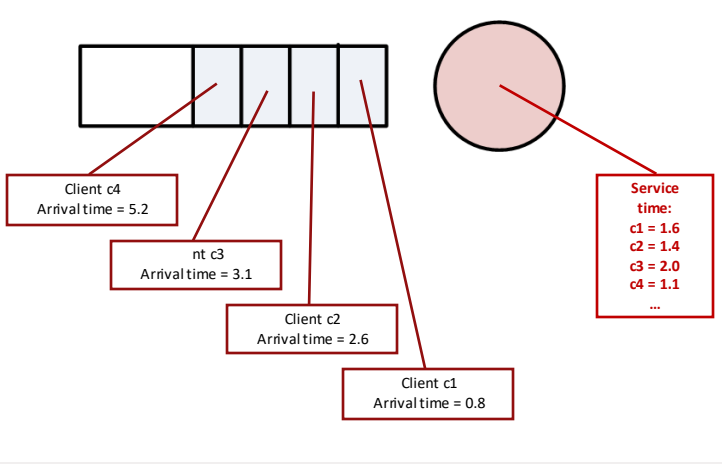
- **Next arrival event:**
  - **c3 at time 3.1**
  - **c4 at time 5.2**

- **Next server event:**
  - **end c2 at time 4.0 (= 2.6 + 1.4)**



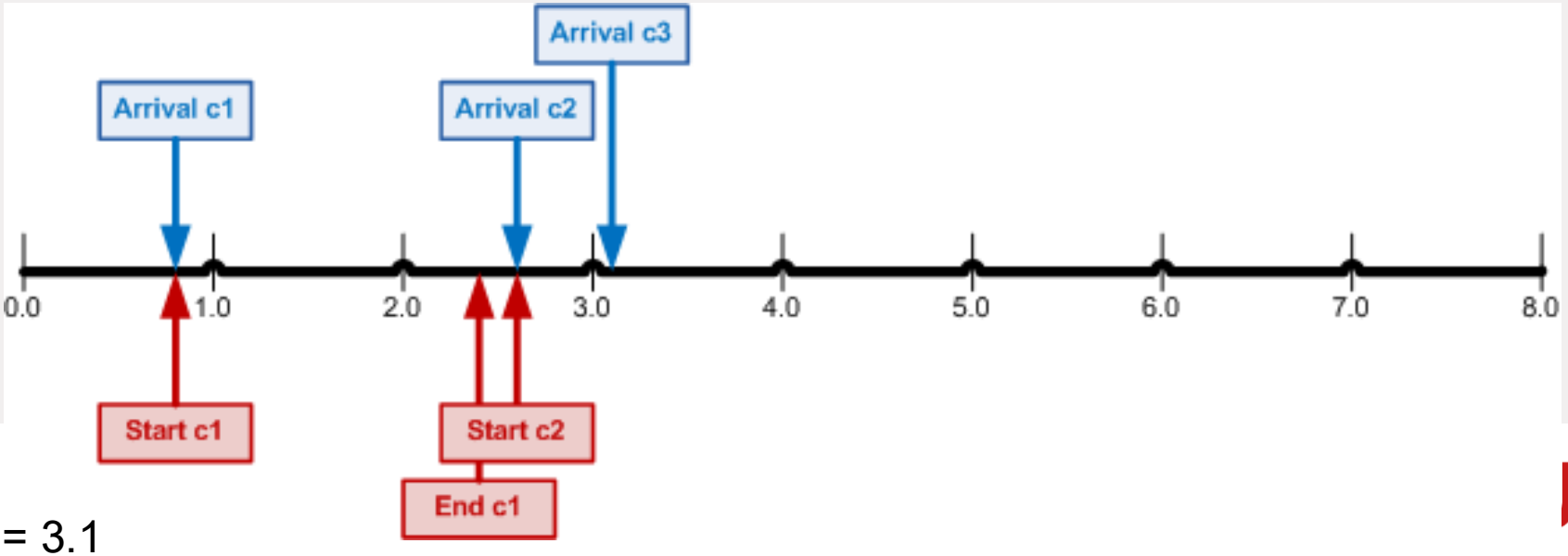
time = 2.6

Time line



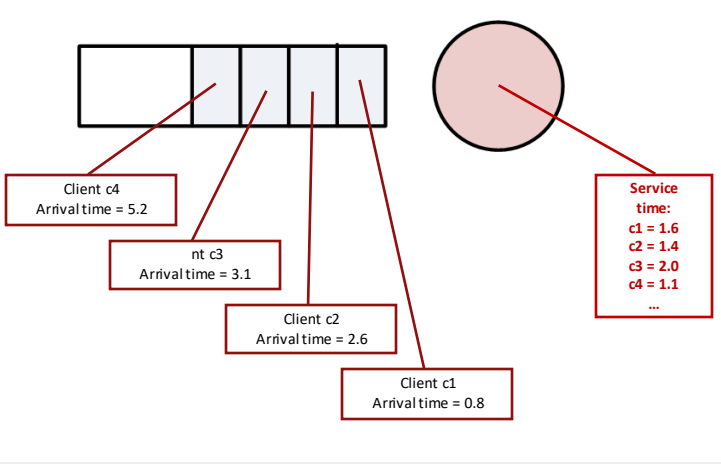
- Next arrival event:
  - c4 at time 5.2

- Next server event:
  - end c2 at time 4.0 (= 2.6 + 1.4)
  - start c3 at time 4.0
  - end c3 at time 6.0 (= 4.0 + 2.0)



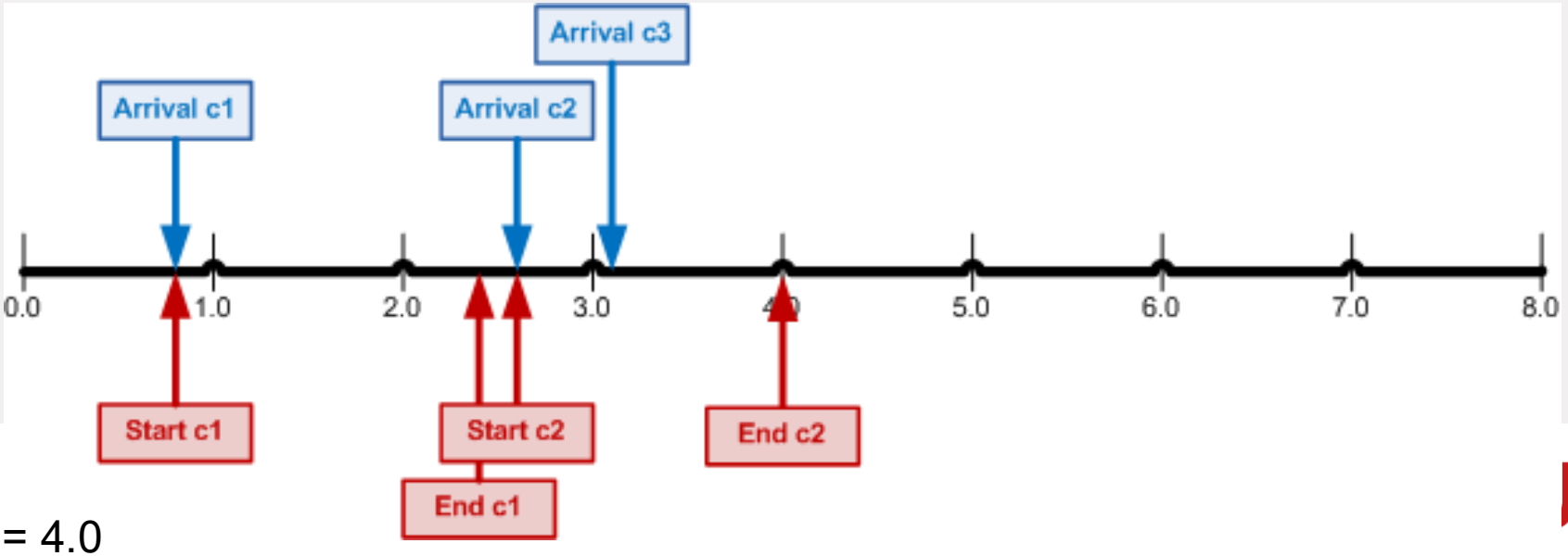
time = 3.1

Time line



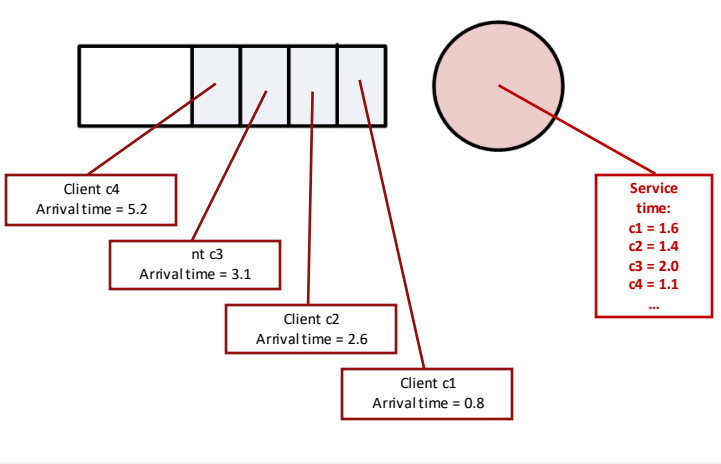
- **Next arrival event:**
  - **c4 at time 5.2**

- **Next server event:**
  - **start c3 at time 4.0**
  - **end c3 at time 6.0 (= 4.0 + 2.0)**



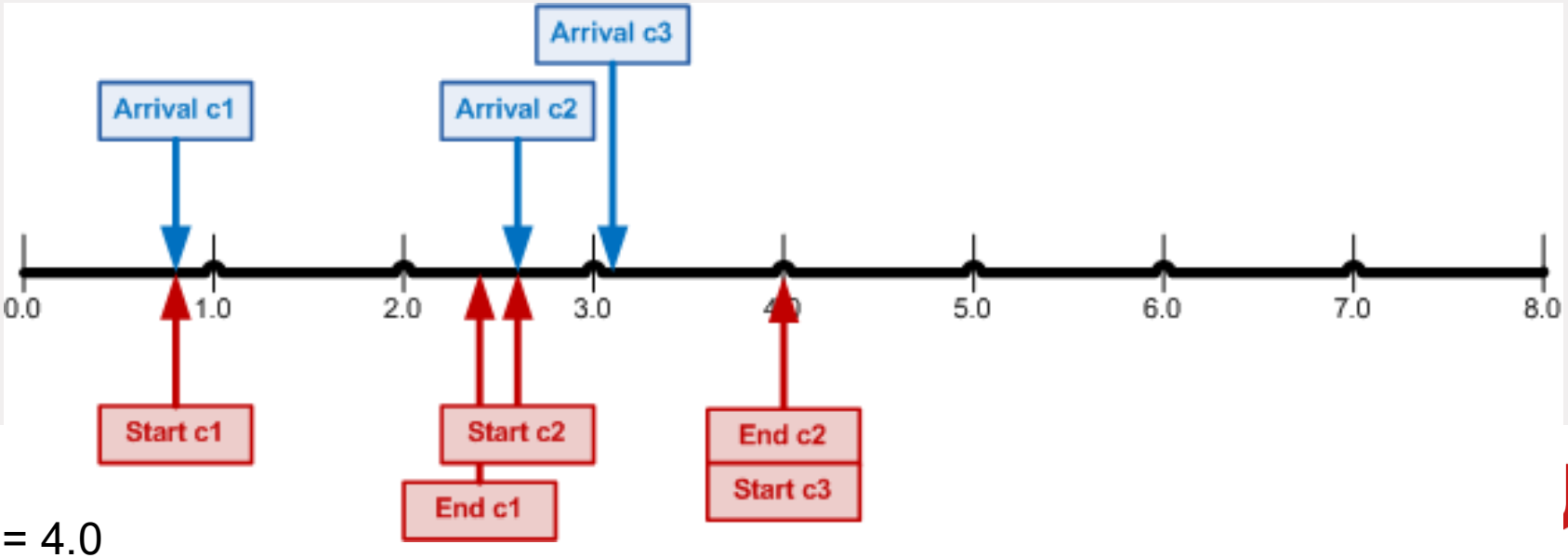
time = 4.0

Time line



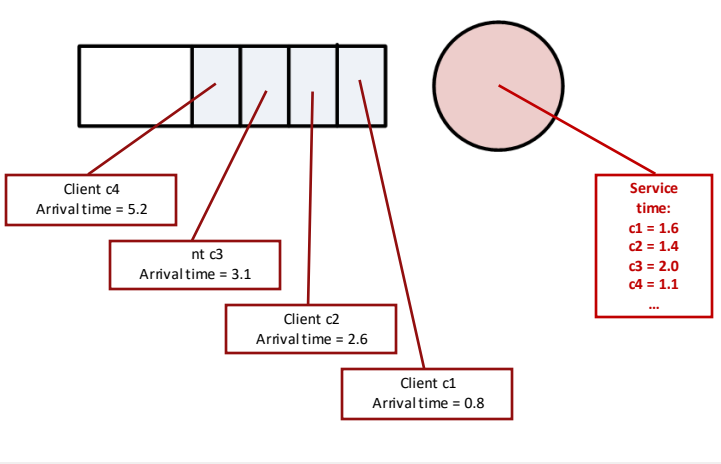
- **Next arrival event:**
  - **c4 at time 5.2**

- **Next server event:**
  - **end c3 at time 6.0 (= 4.0 + 2.0)**



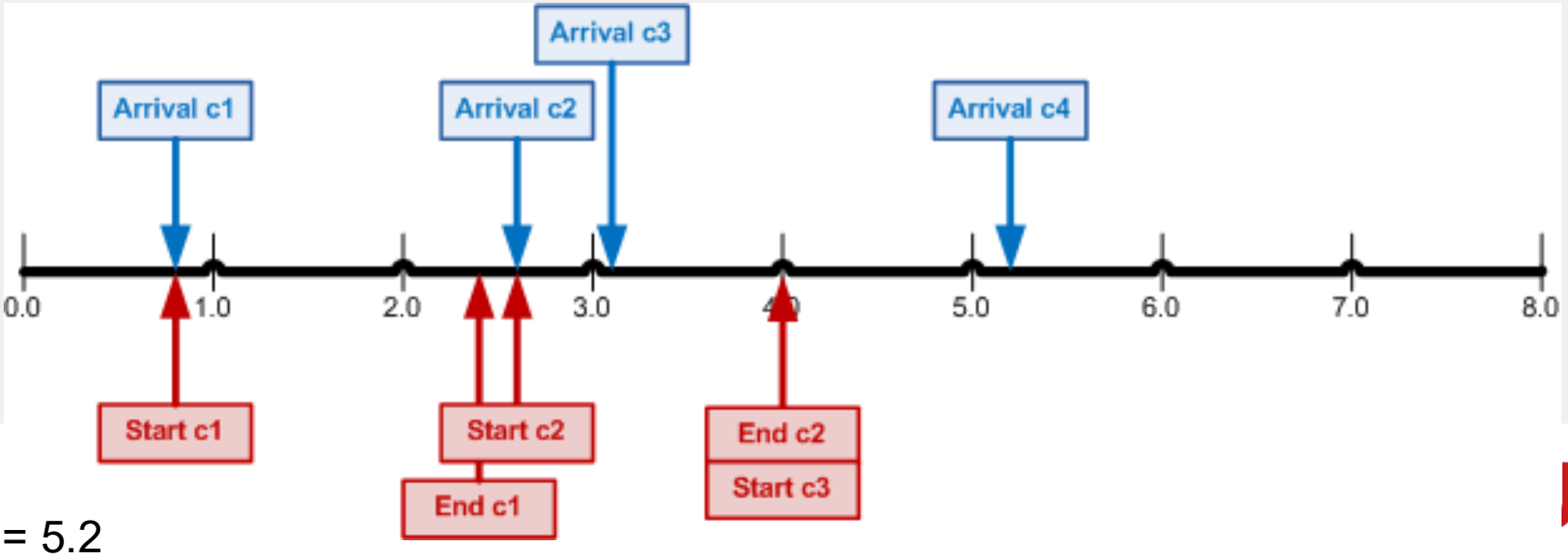
time = 4.0

Time line



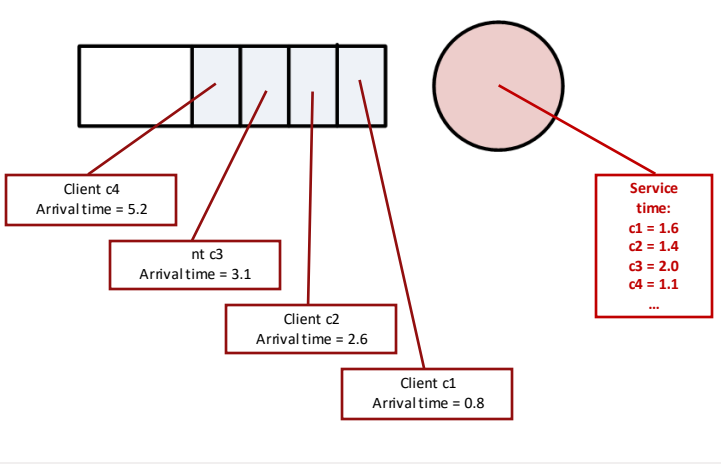
- **Next arrival event:**
  - -

- **Next server event:**
  - **end c3 at time 6.0 (= 4.0 + 2.0)**
  - **start c4 at time 6.0**
  - **end c4 at time 7.1 (= 6.0 + 1.1)**



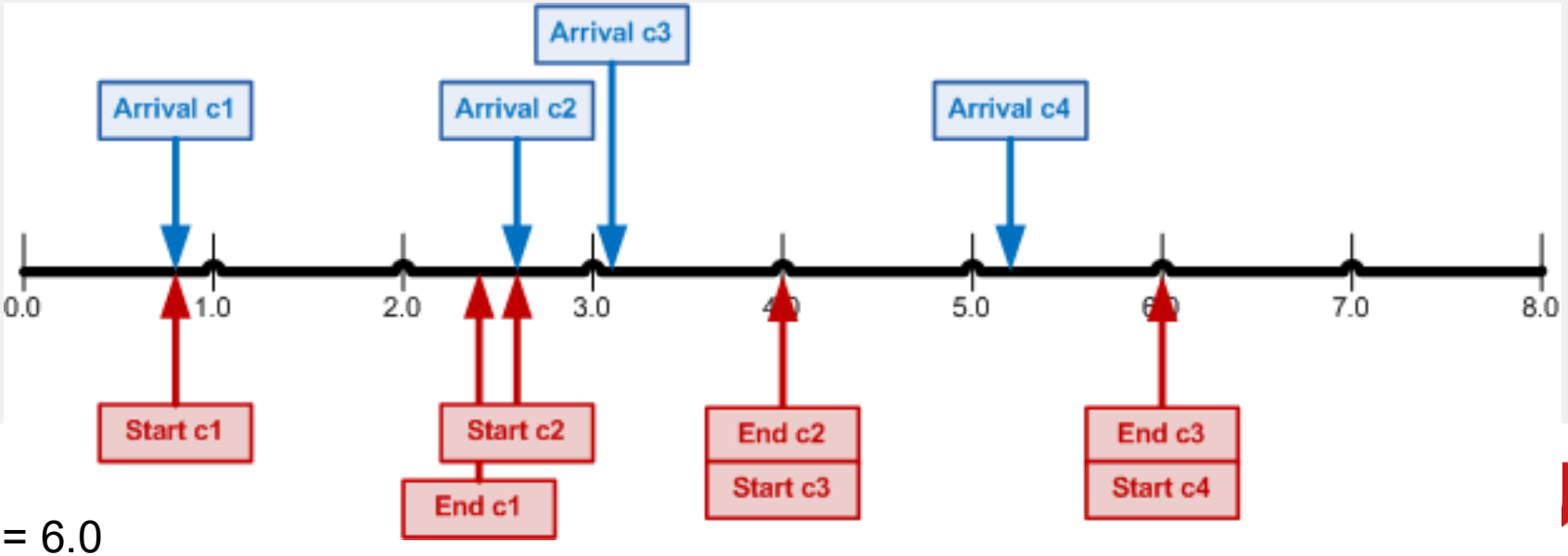
time = 5.2

Time line



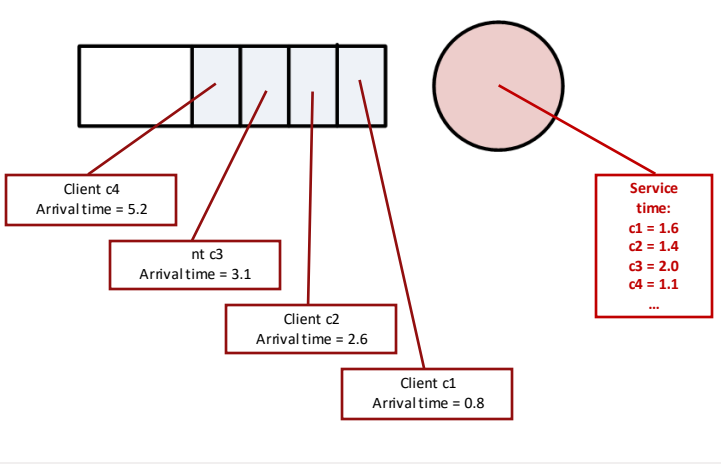
- **Next arrival event:**
  - -

- **Next server event:**
  - **end c4 at time 7.1 (= 6.0 + 1.1)**



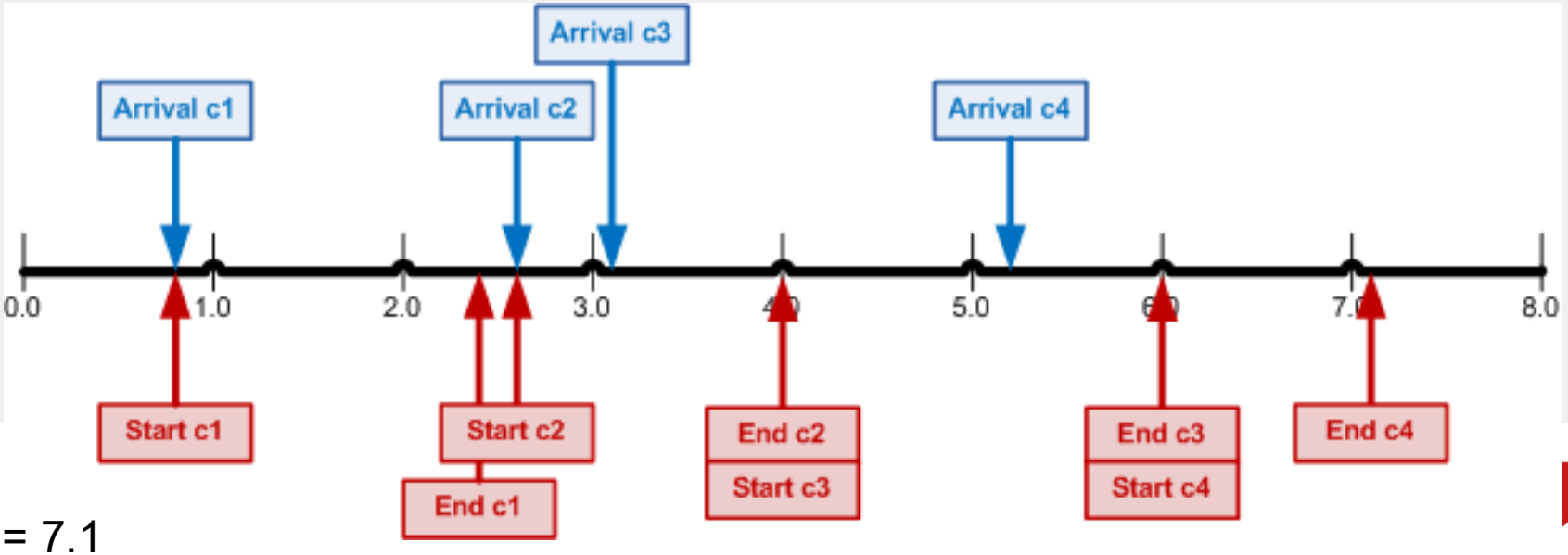
time = 6.0

Time line



- **Next arrival event:**
  - -

- **Next server event:**
  - -



time = 7.1

1BK20 Business Process Simulation

1BK20 Lecture 3e – Petrol station example and manual simulation

Laura Genga

Department of Industrial Engineering and Innovation Sciences, Information Systems Group

Overview on lecture modules

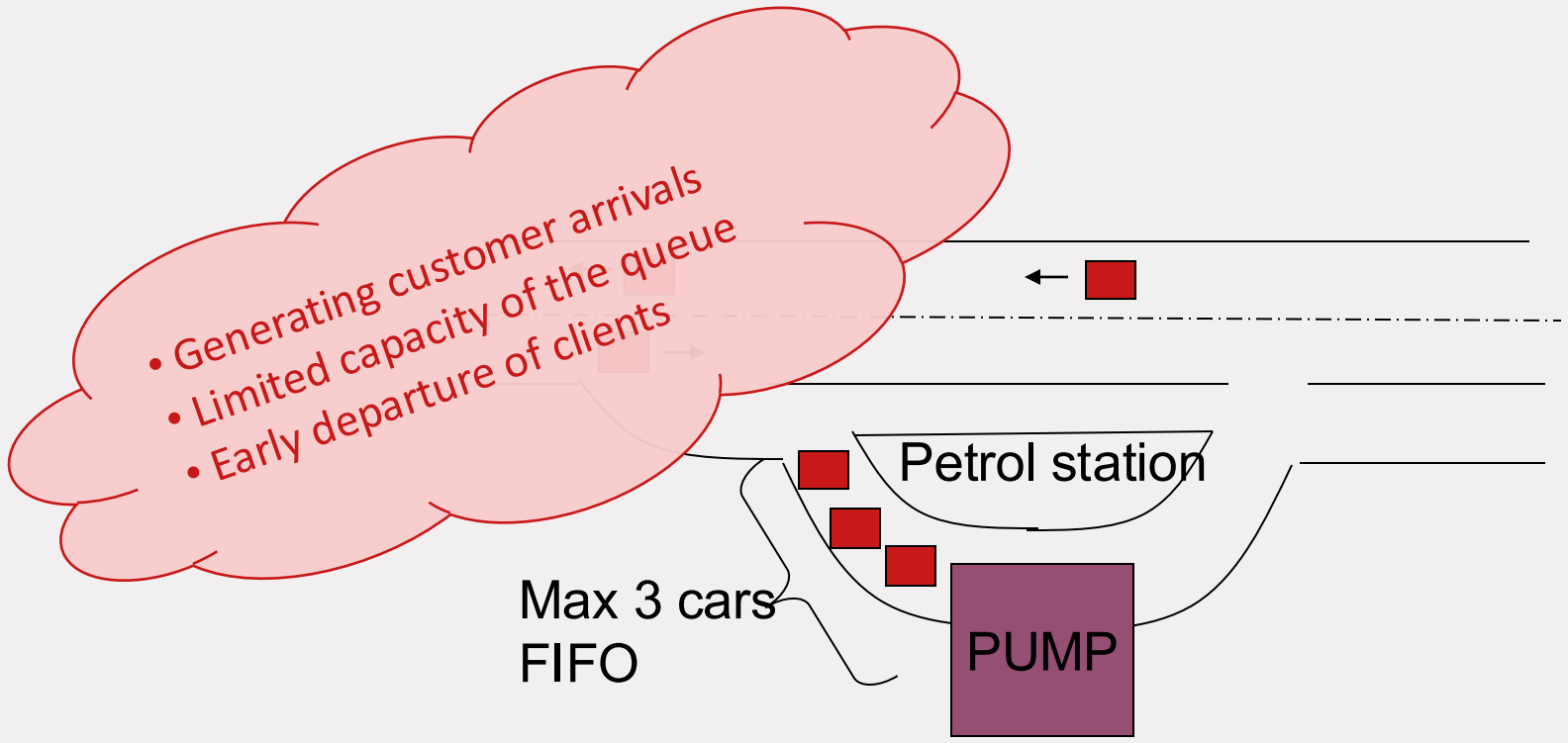
a) Conceptual model: main elements
b) Simple queuing system: process and information model
c) Simple queuing system: transition specifications
d) How does simulation work?
e) Petrol station example and manual simulation
f) Additional examples

Queuing systems in general

Properties:

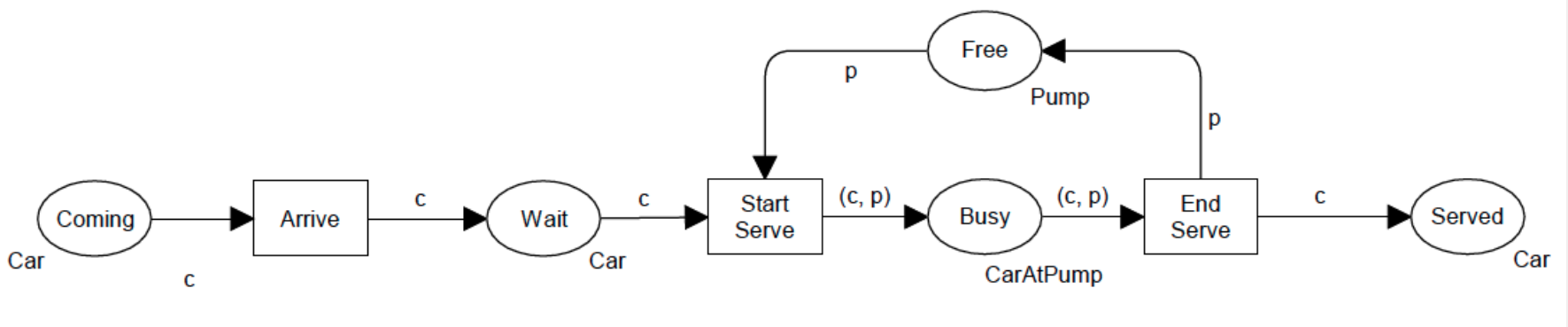
- Generating arrivals
- Early departure of clients
- Limited capacity of the queue
- Maximum number of clients
- Several servers
- Several queues
- Queueing discipline (FIFO, SPT, …)
- Etc.

EXAMPLE: The Petrol Station

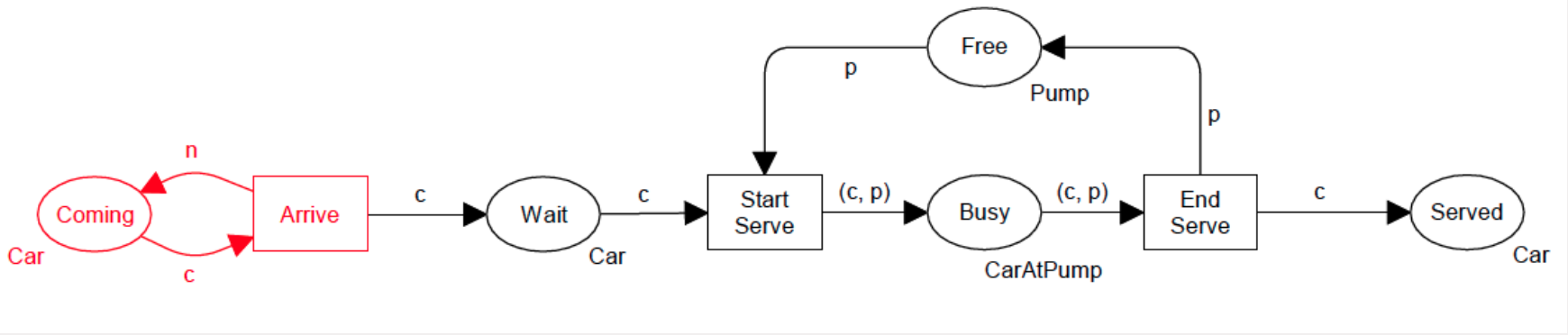


The owner of the petrol station has the feeling that some potential clients are leaving the station because there is no place to wait for service

The process model

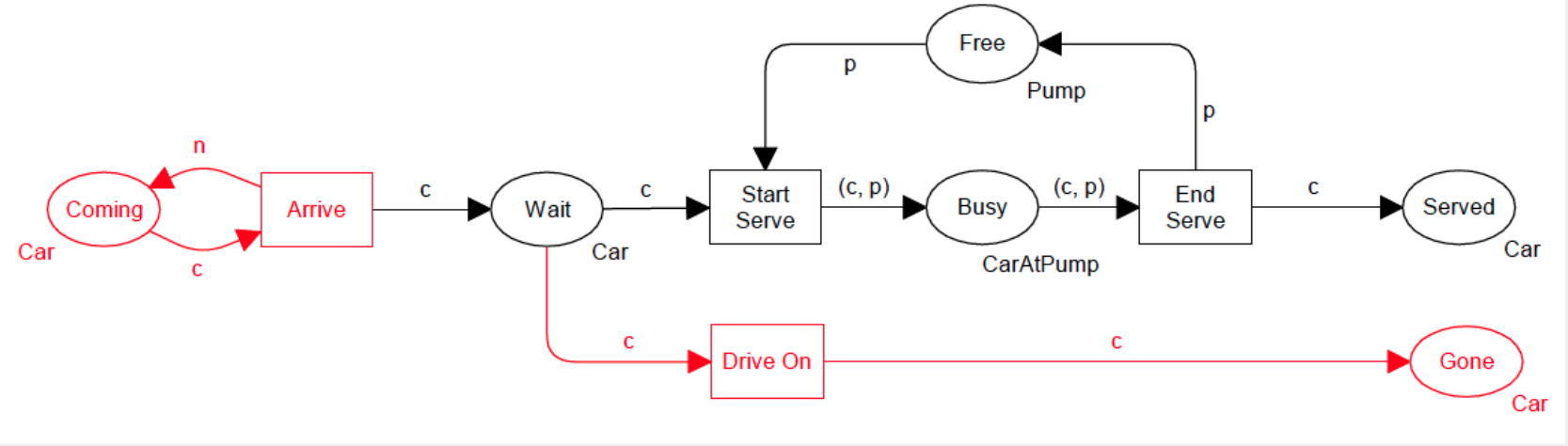


The process model



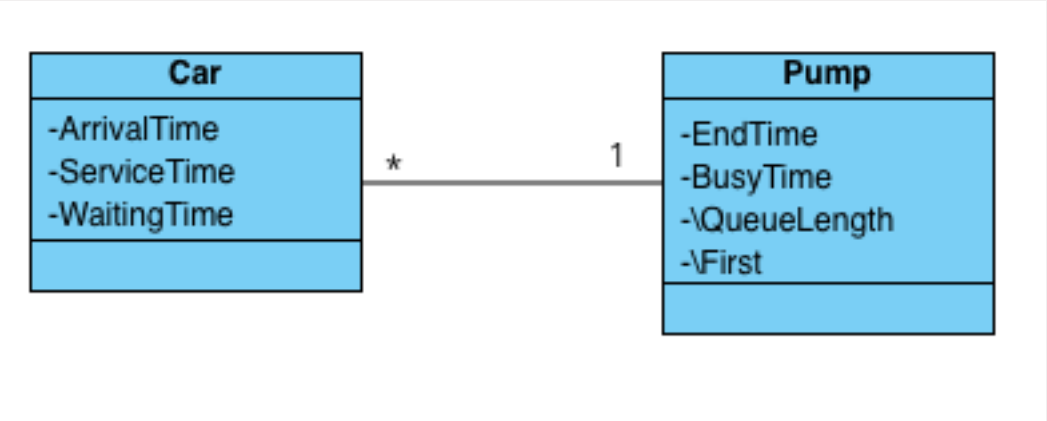
- **Generating customer arrivals**

The process model

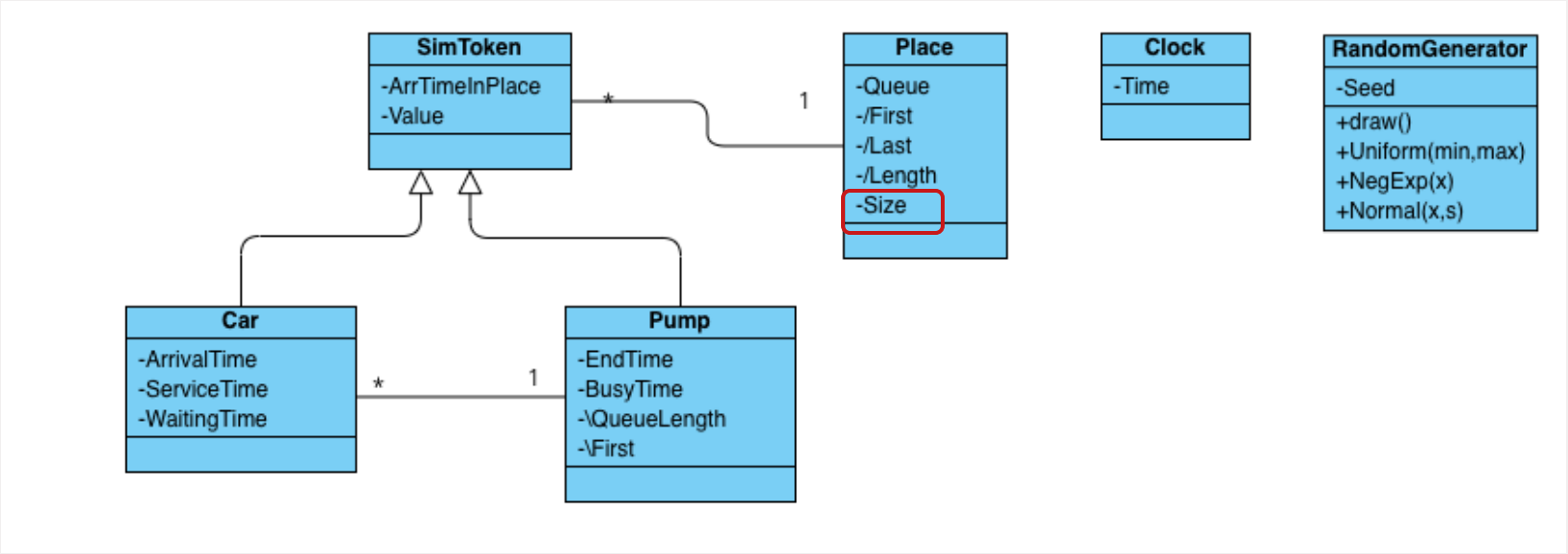


- Generating customer arrivals
- Limited capacity of queue ‘Wait’
- Decision to drive on when queue full

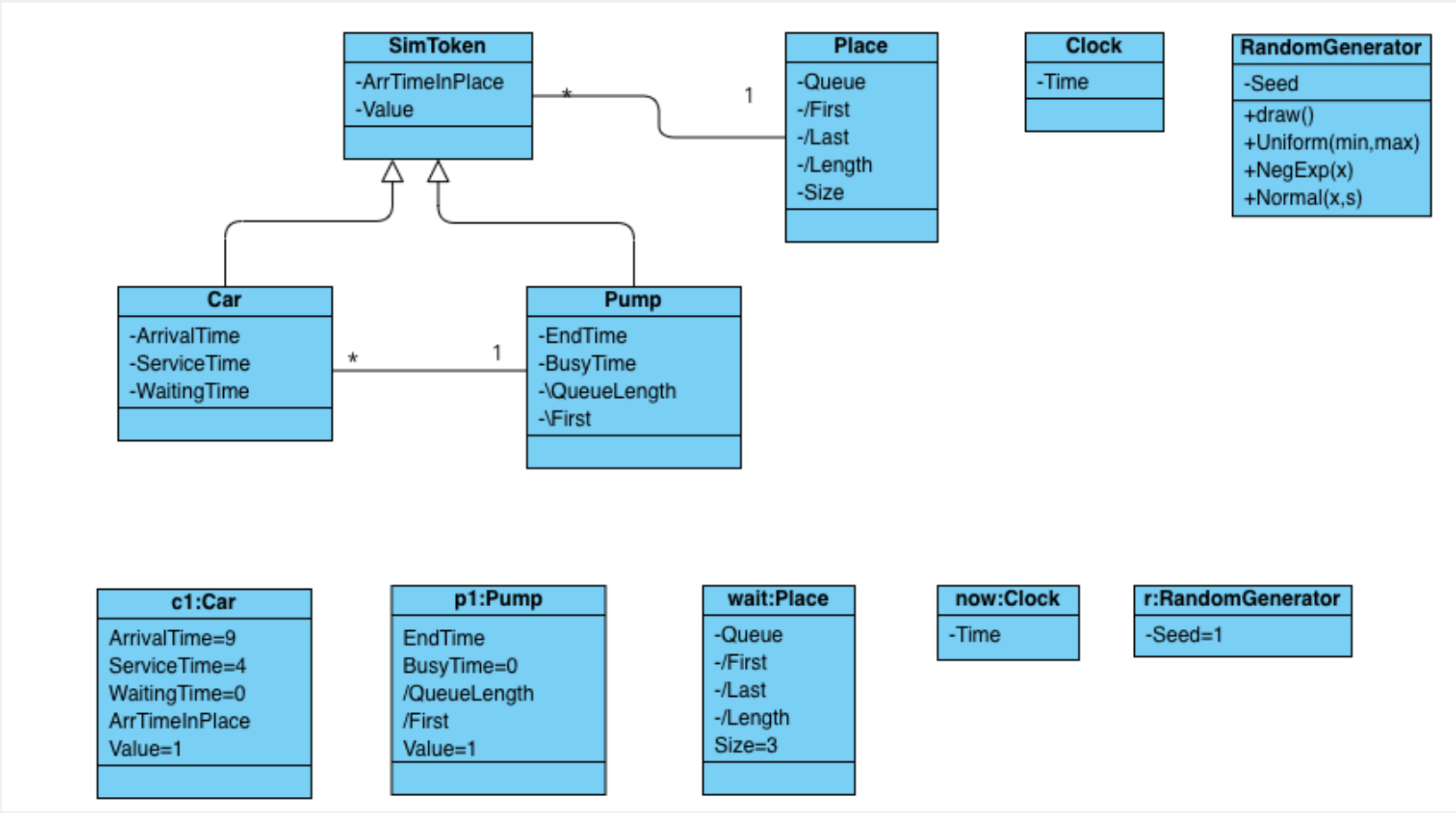
The object model



The object model



Specification of the initial situation



Specification of the initial situation

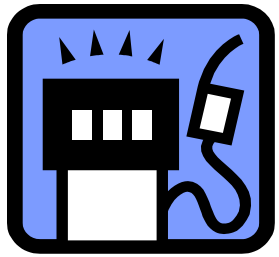
p1: Pump ∧
p1.BusyTime := 0 ∧

c1: Car ∧
c1.Place := Coming ∧
c1.ArrivalTime := r.NegExpo(4) ∧
c1.ServiceTime := r.Uniform(1,6) ∧
c1.WaitingTime := 0 ∧

Wait: Place ∧
Wait.Size = 3 ∧
….

now: Clock ∧ now.Time = 0 ∧
r: RandomGenerator ∧ r.Seed = 1

The transition specifications

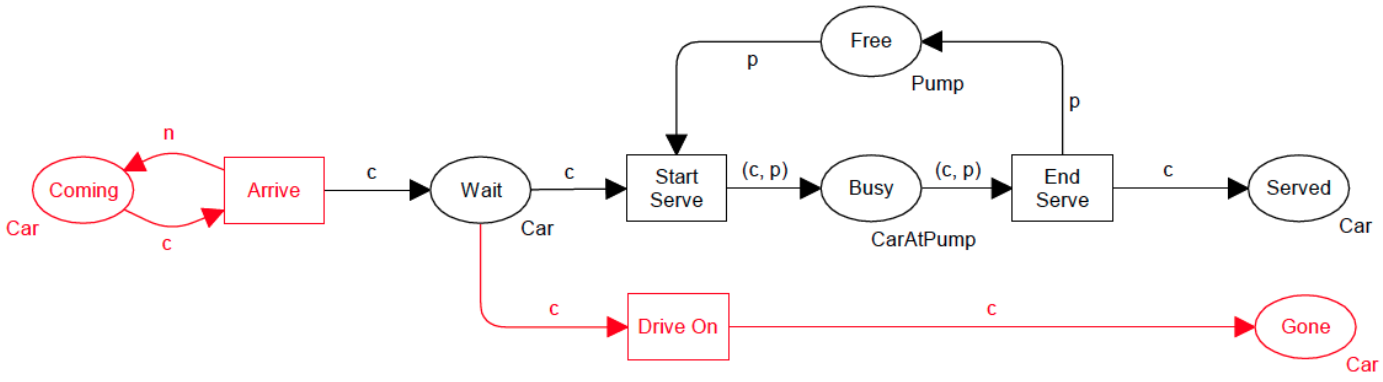


Arrive
Pre: c.ArrivalTime = now.Time
Post: Wait :=Wait.add(c) ∧
Coming:= Coming.add(n) ∧
n.ArrivalTime := c.ArrivalTime + r.NegExpo(4) ∧
n.ServiceTime := r.Uniform(1,6) ∧
n.WaitingTime := 0

StartServe
Pre: c in Wait.First ∧ Wait.Length ≤ Wait.Size;
Post: Busy := Busy.add(c, p) ∧
p.EndTime := now.Time + c.ServiceTime ∧
p.BusyTime := p.BusyTime + c.ServiceTime ∧
c.WaitTime := now.Time – c.ArrivalTime;

EndServe
Pre now.Time = p.EndTime
Post Free:= Free.add(p) ∧
Served:= Served.add(c);

DriveOn
Pre Wait.Length > Wait.Size
∧ c in Wait.Last
Post Gone:= Gone.add(c);



Functions Specifications

Function

PercUnServed =
    (Gone.Length / (Gone.Length + Served.Length)) * 100;

AvgWaitTime =
    Sum(c.WaitTime: c in Served) / Served.Length;

Manual simulation

Replay a small simulation by hand

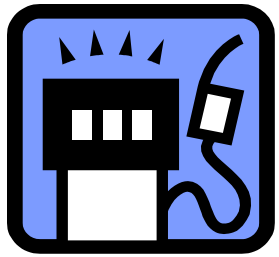
- e.g. for 10 clients

Generate (inter)arrival times and service times

Note down the state of the system after each simulation step, include:

- marking of places with tokens
- relevant values of variables

EXAMPLE: Petrol Station



Suppose we know the first 10 cars

Interarrrival times, services times according to table:

Car	Inter Arrival time	Service time
c1	9	4
c2	5	5
c3	1	6
c4	2	3
c5	1	2
c6	2	1
c7	1	4
c8	5	2
c9	1	1
c10	2	1

TU/e

Manual simulation

now.Time	car	Transition	p.EndTime	Wait	c.WaitTime	Busy	Free	Served	Gone
9	c1	arrive		c1			p1		
9	c1	startserve	13		0	(c1, p1)			
13	c1	endserve	13				p1	c1	
14	c2	arrive		c2			p1	c1	
14	c2	startserve	19		0	(c2, p1)		c1	
15	c3	arrive	19	c3		(c2, p1)		c1	
17	c4	arrive	19	c3, c4		(c2, p1)		c1	
18	c5	arrive	19	c3, c4, c5		(c2, p1)		c1	
19	c2	endserve	19	c3, c4, c5			p1	c1, c2	
19	c3	startserve	25	c4, c5	4	(c3, p1)		c1, c2	
20	c6	arrive	25	c4, c5, c6		(c3, p1)		c1, c2	
21	c7	arrive	25	c4, c5, c6		(c3, p1)		c1, c2	c7
25	c3	endserve		c4, c5, c6			p1	c1, c2, c3	c7
25	c4	startserve	28	c5, c6	8	(c4, p1)		c1, c2, c3	c7
26	c8	arrive	28	c5, c6, c8		(c4, p1)		c1, c2, c3	c7
27	c9	arrive	28	c5, c6, c8		(c4, p1)		c1, c2, c3	c7, c9
28	c4	endserve	28	c5, c6, c8			p1	c1, c2, c3, c4	c7, c9
28	c5	startserve	30	c6, c8	10	(c5, p1)		c1, c2, c3, c4	c7, c9
29	c10	arrive	30	c6, c8, c10		(c5, p1)		c1, c2, c3, c4	c7, c9
30	c5	endserve	30	c6, c8, c10			p1	c1, c2, c3, c4, c5	c7, c9
30	c6	startserve	31	c8, c10	10	(c6, p1)		c1, c2, c3, c4, c5	c7, c9
31	c6	endserve	31	c8, c10			p1	c1, c2, c3, c4, c5, c6	c7, c9
31	c8	startserve	33	c10	5	(c8, p1)		c1, c2, c3, c4, c5, c6	c7, c9
33	c8	endserve	33	c10			p1	c1, c2, c3, c4, c5, c6, c8	c7, c9
33	c10	startserve	34		4	(c10, p1)		c1, c2, c3, c4, c5, c6, c8	c7, c9
34	c10	endserve	34				p1	c1, c2, c3, c4, c5, c6, c8, c10	c7, c9
Length								8	2
Sum					41				

TU/e

Manual simulation

now.Time	car	Transition	p.EndTime	Wait	c.WaitTime	Busy	Free	Served	Gone
9	c1	arrive		c1			p1		
9	c1	startserve	13		0	(c1, p1)			
13	c1	endserve	13				p1	c1	
14	c2	arrive		c2			p1	c1	
14	c2	startserve	19		0	(c2, p1)		c1	
15	c3	arrive	19	c3		(c2, p1)		c1	
17	c4	arrive	19	c3, c4		(c2, p1)		c1	
18	c5	arrive	19	c3, c4, c5		(c2, p1)		c1	
19	c2	endserve	19	c3, c4, c5			p1	c1, c2	
19	c3	startserve	25	c4, c5	4	(c3, p1)		c1, c2	
20	c6	arrive	25	c4, c5, c6		(c3, p1)		c1, c2	
21	c7	arrive	25	c4, c5, c6		(c3, p1)		c1, c2	c7
25	c3	endserve		c4, c5, c6			p1	c1, c2, c3	c7
25	c4	startserve	28	c5, c6	8	(c4, p1)		c1, c2, c3	c7
26	c8	arrive	28	c5, c6, c8		(c4, p1)		c1, c2, c3	c7
27	c9	arrive	28	c5, c6, c8		(c4, p1)		c1, c2, c3	c7, c9
28	c4	endserve	28	c5, c6, c8			p1	c1, c2, c3, c4	c7, c9
28	c5	startserve	30	c6, c8	10	(c5, p1)		c1, c2, c3, c4	c7, c9
29	c10	arrive	30	c6, c8, c10		(c5, p1)		c1, c2, c3, c4	c7, c9
30	c5	endserve	30	c6, c8, c10			p1	c1, c2, c3, c4, c5	c7, c9
30	c6	startserve	31	c8, c10	10	(c6, p1)		c1, c2, c3, c4, c5	c7, c9
31	c6	endserve	31	c8, c10			p1	c1, c2, c3, c4, c5, c6	c7, c9
31	c8	startserve	33	c10	5	(c8, p1)		c1, c2, c3, c4, c5, c6	c7, c9
33	c8	endserve	33	c10			p1	c1, c2, c3, c4, c5, c6, c8	c7, c9
33	c10	startserve	34		4	(c10, p1)		c1, c2, c3, c4, c5, c6, c8	c7, c9
34	c10	endserve	34				p1	c1, c2, c3, c4, c5, c6, c8, c10	c7, c9
Length								8	2
Sum				41					

TU/e

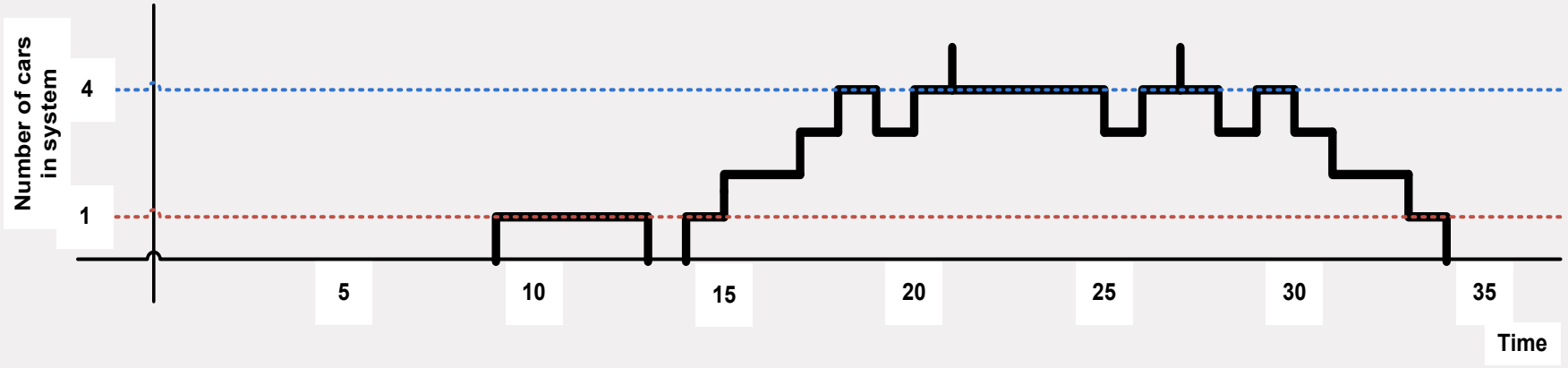
Manual simulation

Percentage of customers driving on:

PrecUnServed =

(Gone.Length / (Gone.Length + Served.Length)) * 100%

2/(8+2) * 100% = 20%



1BK20 Business Process Simulation

1BK20 Lecture 3f – Additional examples

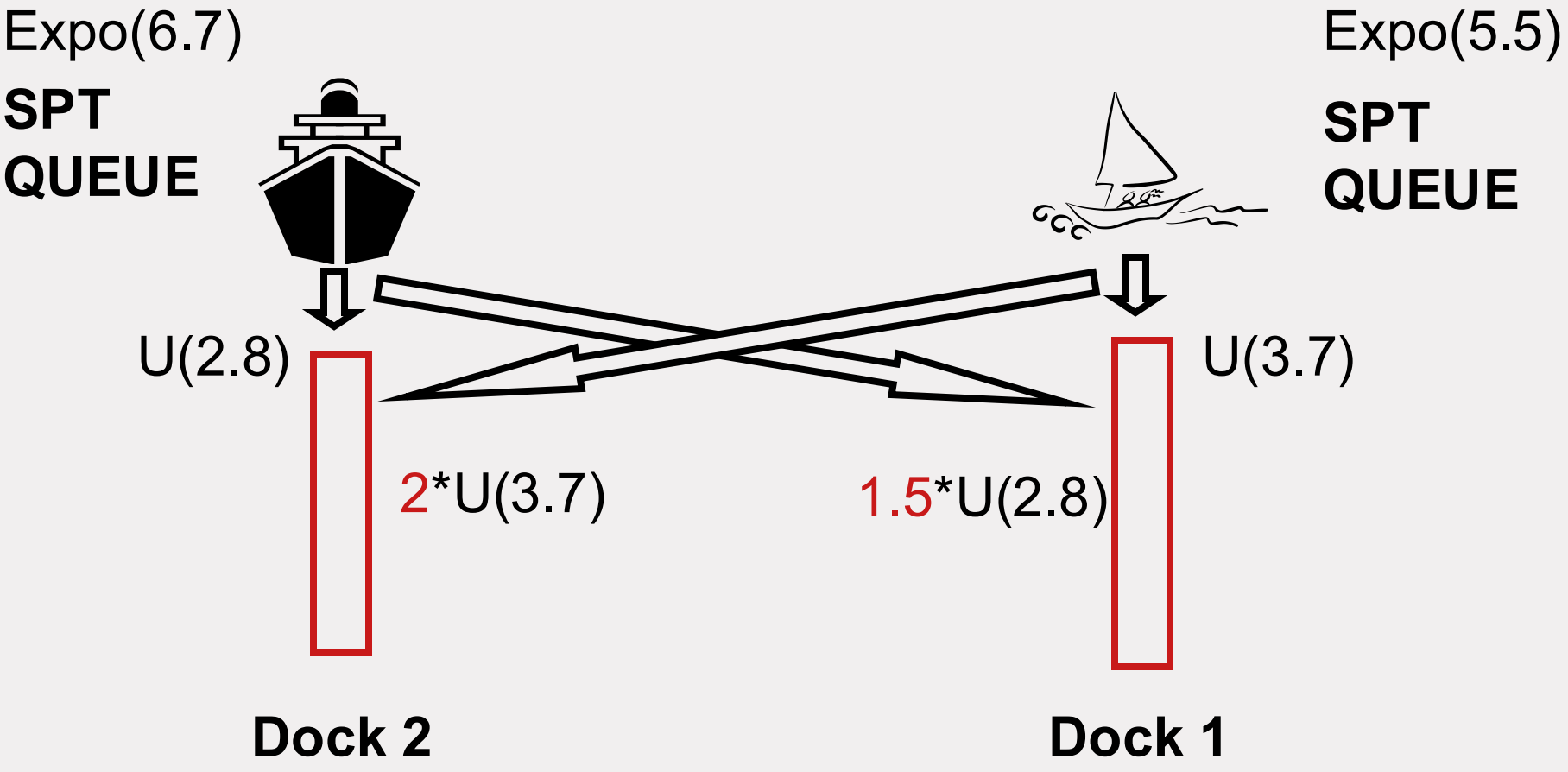
Laura Genga

Department of Industrial Engineering and Innovation Sciences, Information Systems Group

Overview on lecture modules

a) Conceptual model: main elements
b) Simple queuing system: process and information model
c) Simple queuing system: transition specifications
d) How does simulation work?
e) Petrol station example and manual simulation
f) Additional examples

EXAMPLE 2: The Harbour Case



The decision problem

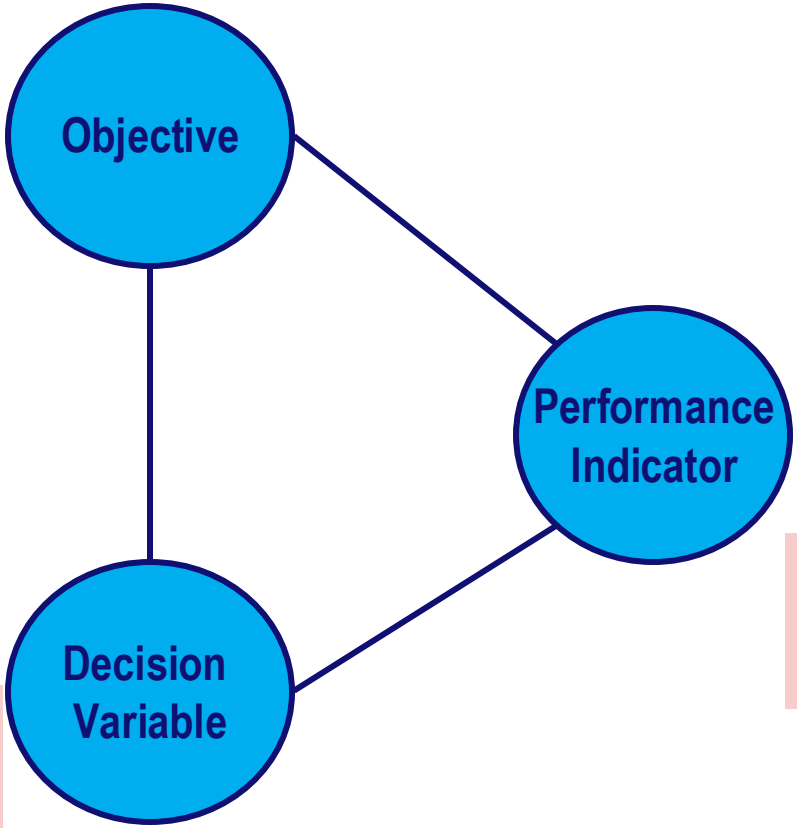
The management team of the harbour wonders what is the impact on the mean expected throughput time of:

closing dock1 to big ships and dock2 to small ships?
using a FIFO rule (First In First Out) instead of the SPT (Shortest Process Time) rule?

Can we help the management team to get answers to their questions?

Step 1.1: Decision Frame

Decrease “mean expected throughput time”



“mean expected throughput time”

- queuing discipline
- resource allocation strategy

TU/e

Step 1.2: Research Questions

What is the mean expected throughput time of ships at the harbour?

- In the current case
- if we close dock1 to big ships and dock2 to small ships,
- if we use a FIFO rule instead of the SPT rule
- if we make both adjustments

In which of these scenario's does the average expected throughput time decrease?

Step 1.3: Scope and level of detail

The two docks with
- Start events
- End events

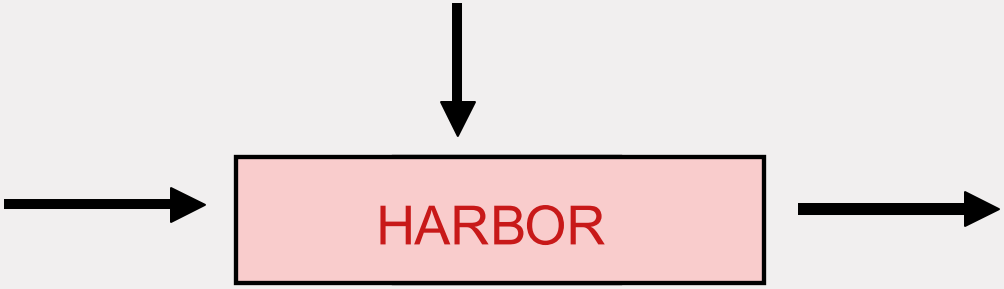
The ships with
- Arrival events
- Change queue events
- Start service events

The queues with
- Their queueing discipline (SPT)

Step 2.1: Black Box

Environmental Variables

- interarrival time of (types of) ships
- service time of docks



- queue discipline (FIFO / SPT)
- dock allocation strategy

- average throughput time

TU/e

Step 2.2: Assumptions and Givens

G1: There are two types of ships: small and big
G2: Interarrival times of

- big ships (Expo(5,5)) and
- small ships (Expo(6,7))

G3: Service times of

- dock 1 (Uniform(3,7)) and
- dock 2 (Uniform(2,8))

A1: Docks operate 24/7
A2: No maintenance of docks required
A3: Only the first ship in the queue (i.e. the one with the shortest processing time, or the first one in the row) may change queues

TU/e

Step 2.3: Is simulation suitable?

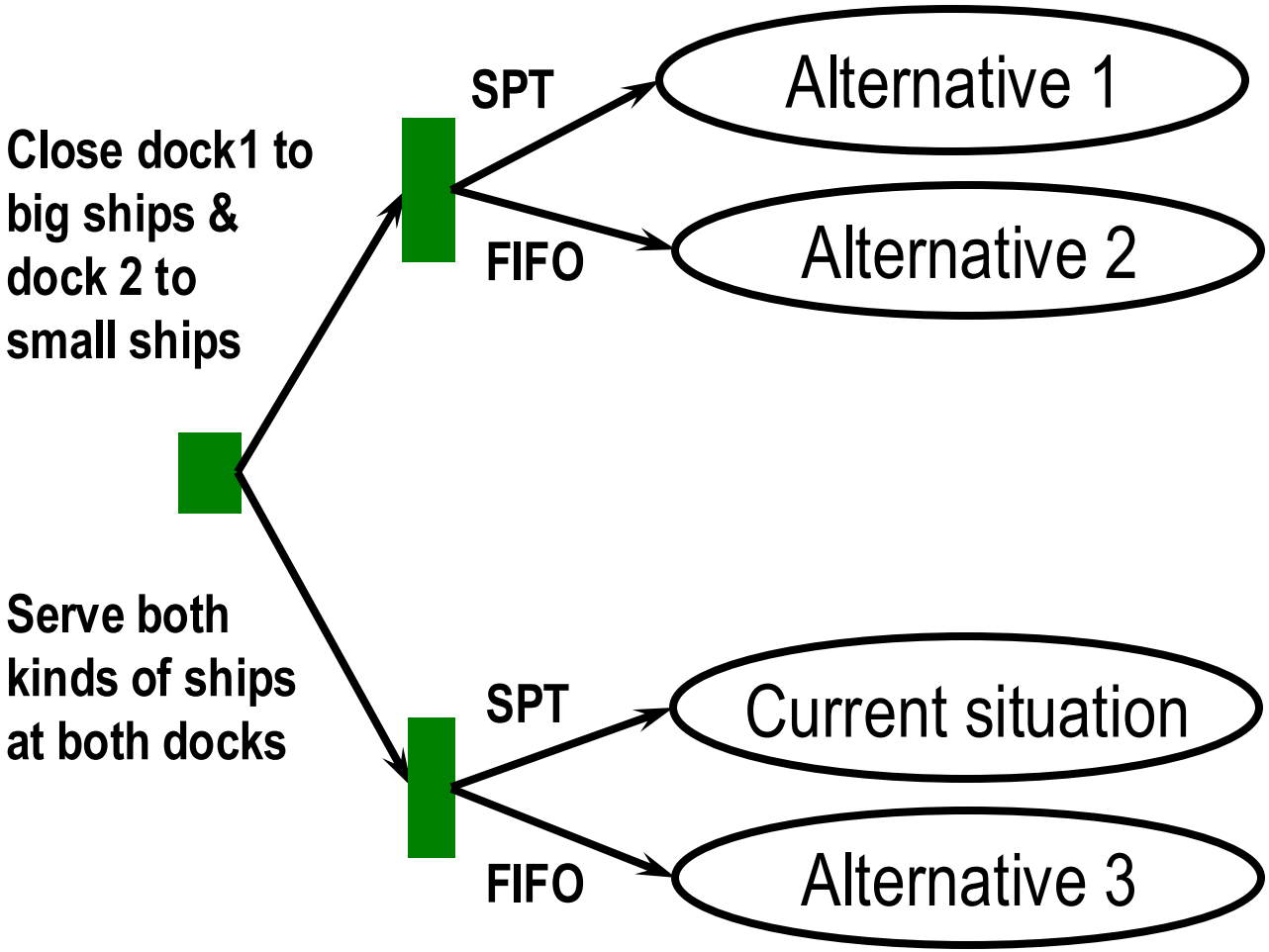
Change of queues can not be modeled in analytical models
Approximation with two M/G/1 queues possible:

- Dock 1 only for small ships, dock 2 only for big ships
- FIFO instead of SPT rule

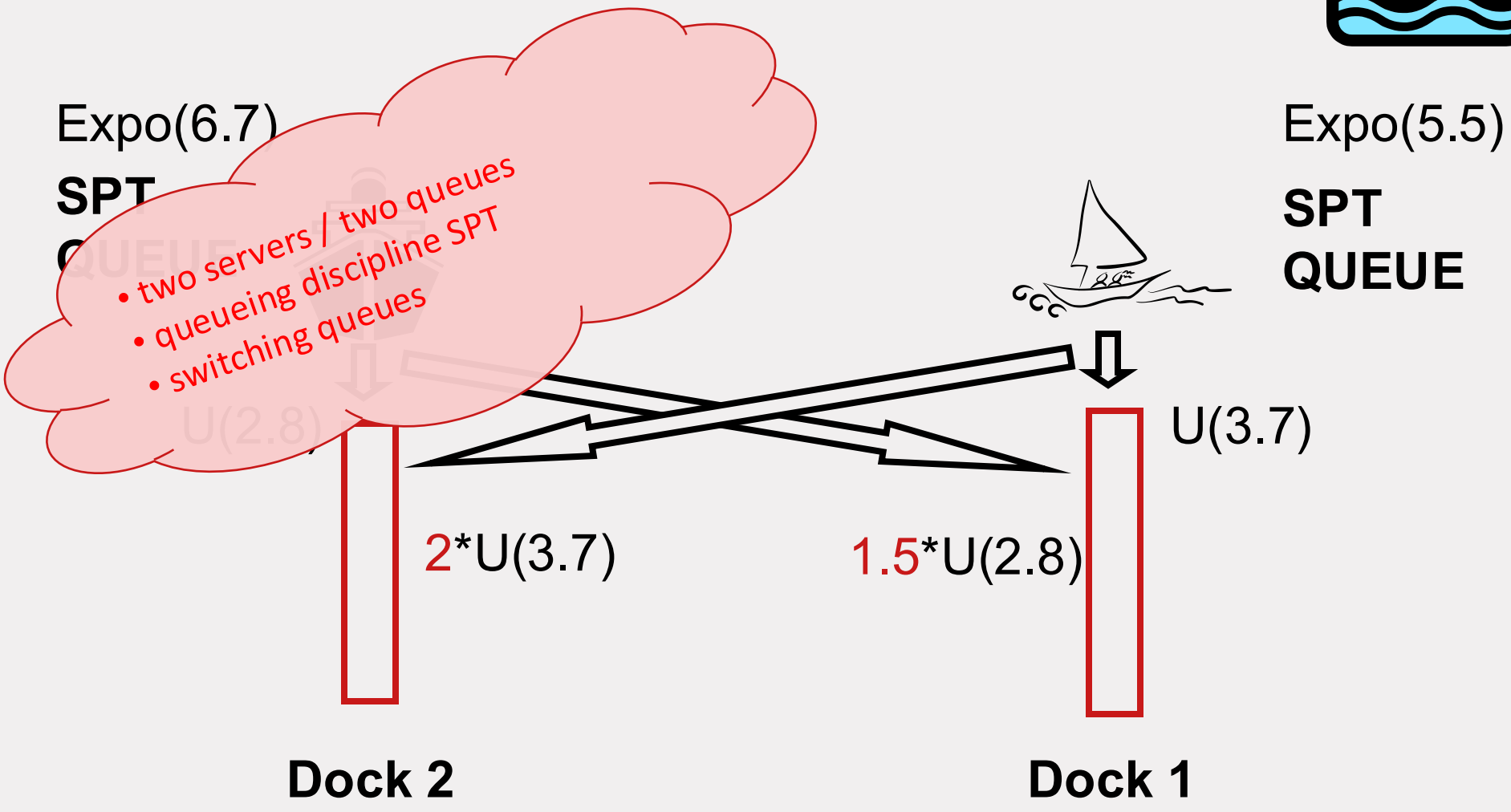
Experimentation is not desirable.

TU/e

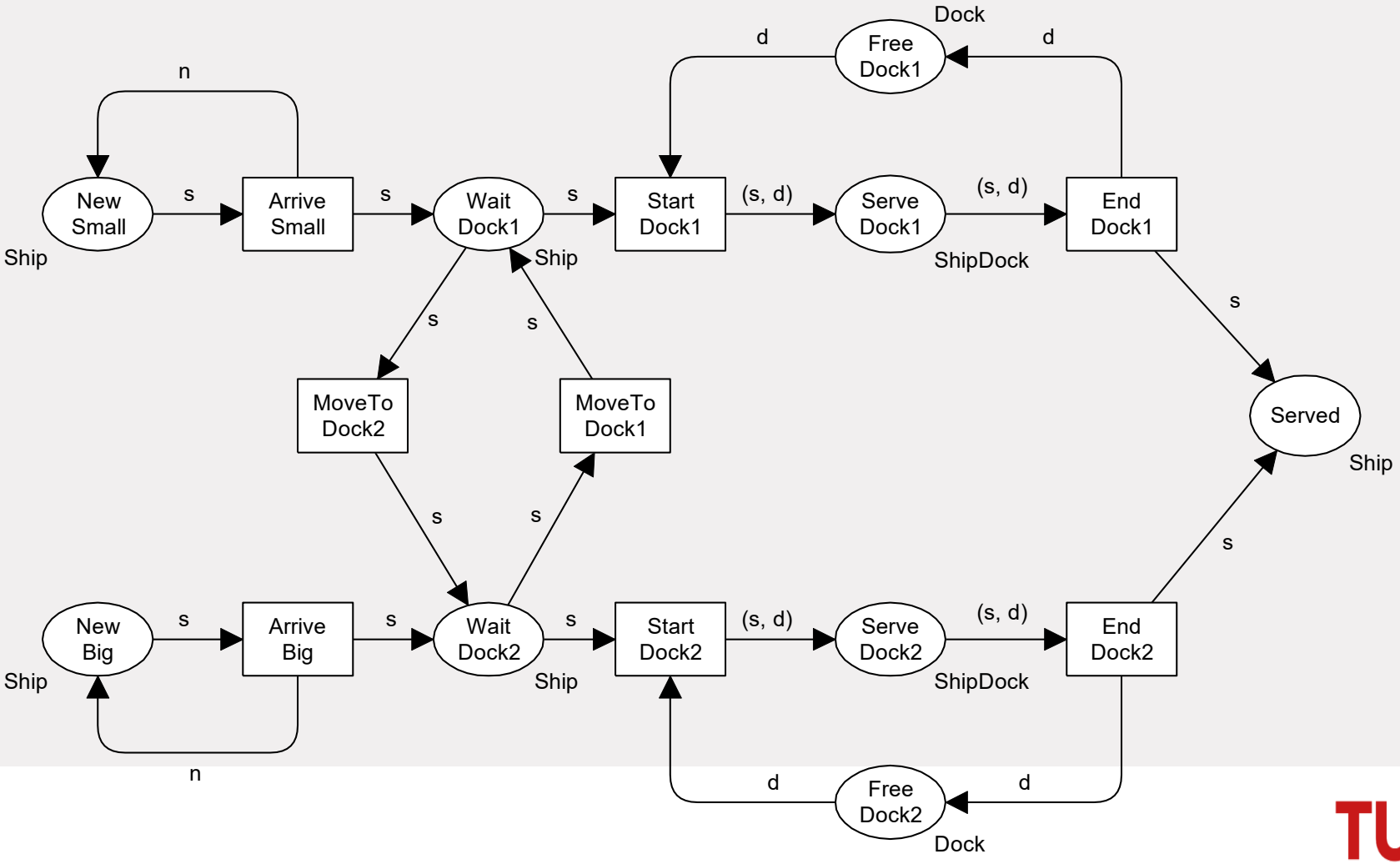
Step 2.4: Number of models



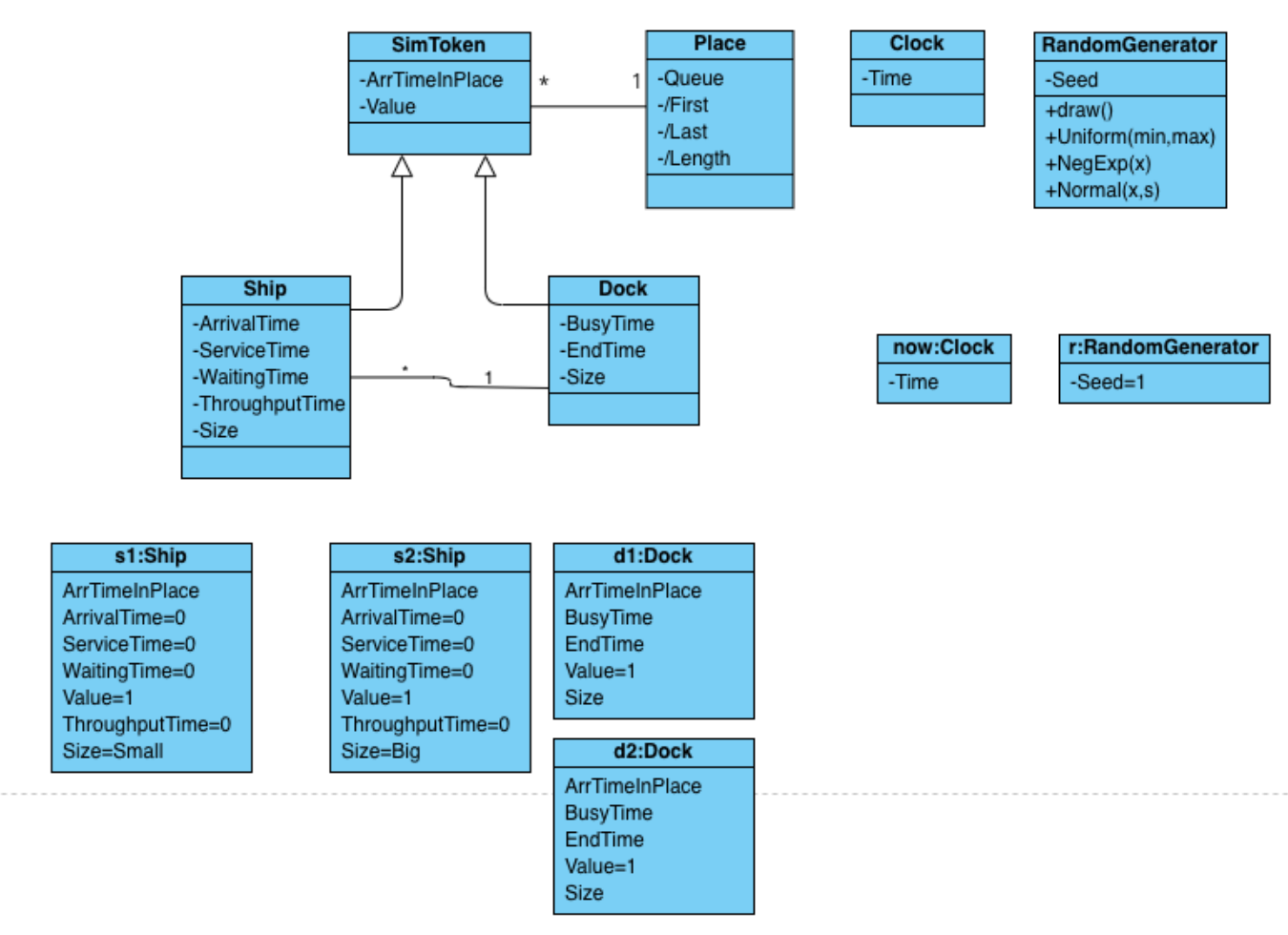
EXAMPLE 2: The Harbour Case



STEP 3: Process model



STEP 3: Information model



STEP 3: Transition specifications

ArriveSmall
Pre s.ArrivalTime = now.Time;
Post WaitDock1:= WaitDock1.add(s) ∧ s.ServiceTime := r.Uniform(3,7) ∧
NewSmall:= NewSmall.add(n) ∧
n.ArrivalTime := now.Time + r.Negexp(5.5);

ArriveBig
Pre s.ArrivalTime = now.Time;
Post WaitDock2:= WaitDock2.add(s) ∧ s.ServiceTime := r.Uniform(2,8) ∧
NewBig := NewBig.add(n) ∧
n.ArrivalTime := now.Time + r.Negexp(6.7);

MoveToDock1
Pre WaitDock1.Length = 0 ∧ d1.Place = FreeDock1 ∧ WaitDock2.Length>0
%Dock 1 should empty and ships should be waiting for Dock 2%
Post WaitDock1= WaitDock1.add(s) ∧
s.ServiceTime := r.Uniform(2,8) * 1.5

MoveToDock2
Pre WaitDock2.Length = 0 ∧ d2.Place = FreeDock2 ∧ WaitDock1.Length>0
%Dock 2 should empty and ships should be waiting for Dock 1%
Post WaitDock2:= WaitDock2.add(s) ∧
s.ServiceTime := r.Uniform(3, 7)*2

StartDock1
Pre s.ServiceTime = Min(s in WaitDock1 .Queue : s.ServiceTime)
%SPT selection%
Post ServeDock1 := ServeDock1.add(s d) ∧
d.EndTime := now.Time + s.ServiceTime ∧
d.BusyTime := d.BusyTime + s.ServiceTime ∧
s.WaitingTime := now.Time - s.ArrivalTime ;

StartDock2
Pre s.ServiceTime = Min(s in WaitDock2.Queue : s.ServiceTime) %SPT selection%
Post ServeDock2 ServeDock2.add(s, d) ∧
d.EndTime := now.Time + s.ServiceTime ∧
d.BusyTime := d.BusyTime+ s.ServiceTime ∧
s.WaitingTime := now.Time - s.ArrivalTime ;

EndDock1
Pre now.Time = d.EndTime
Post Served := Served .add(t) ∧
s.ThroughputTime := now.Time - s.ArrivalTime ∧
FreeDock1 := FreeDock1.add(d);

EndDock2
Pre now.Time = d.EndTime
Post Served := Served.add(t) ∧
s.ThroughputTime := now.Time - s.ArrivalTime ∧
FreeDock2 := FreeDock2.add(d);

TU/e

STEP 3: Init & function specifications

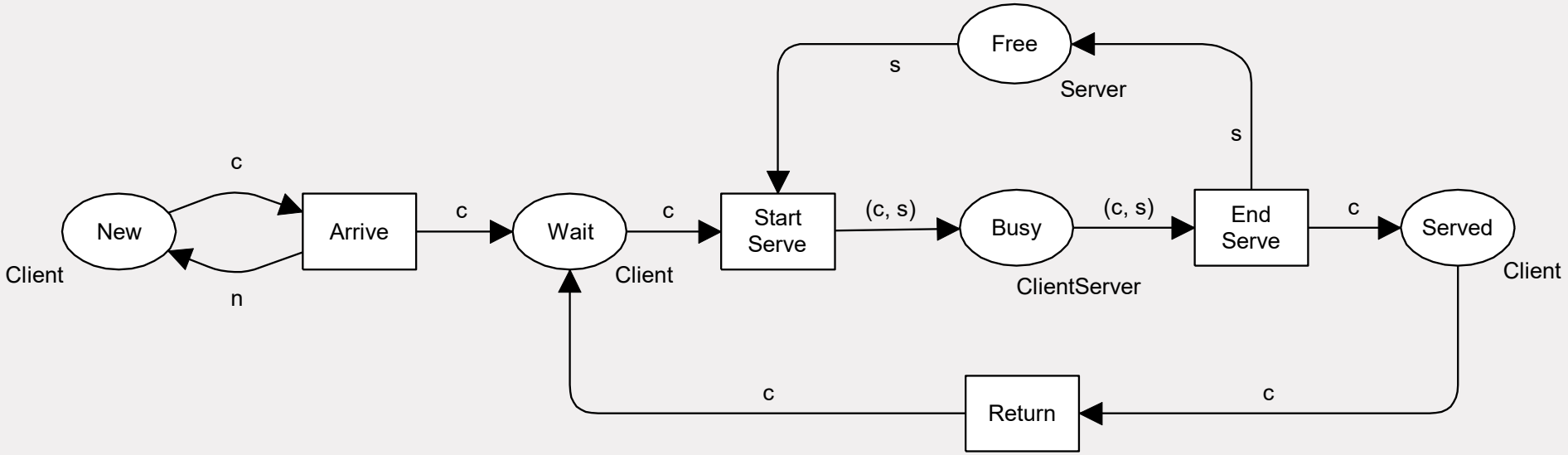
Init
s1: Ship $\wedge$ s1.Place = NewSmall $\wedge$ s1.ArrivalTime = 0 $\wedge$ s1.Size = small $\wedge$
s2: Ship $\wedge$ s2.Place = NewBig $\wedge$ s2.ArrivalTime = 0 $\wedge$ s2.Size = big $\wedge$
d1: Dock $\wedge$ d1.Place = FreeDock1 $\wedge$ d1.BusyTime = 0 $\wedge$
d2: Dock $\wedge$ d2.Place = FreeDock2 $\wedge$ d2.BusyTime = 0 $\wedge$
r: RandomGenerator $\wedge$ now: Clock;

Functions
MeanTPT = Sum(s.ThroughputTime: s in Served) / Served.Length;

TU/e

EXAMPLE: closed system

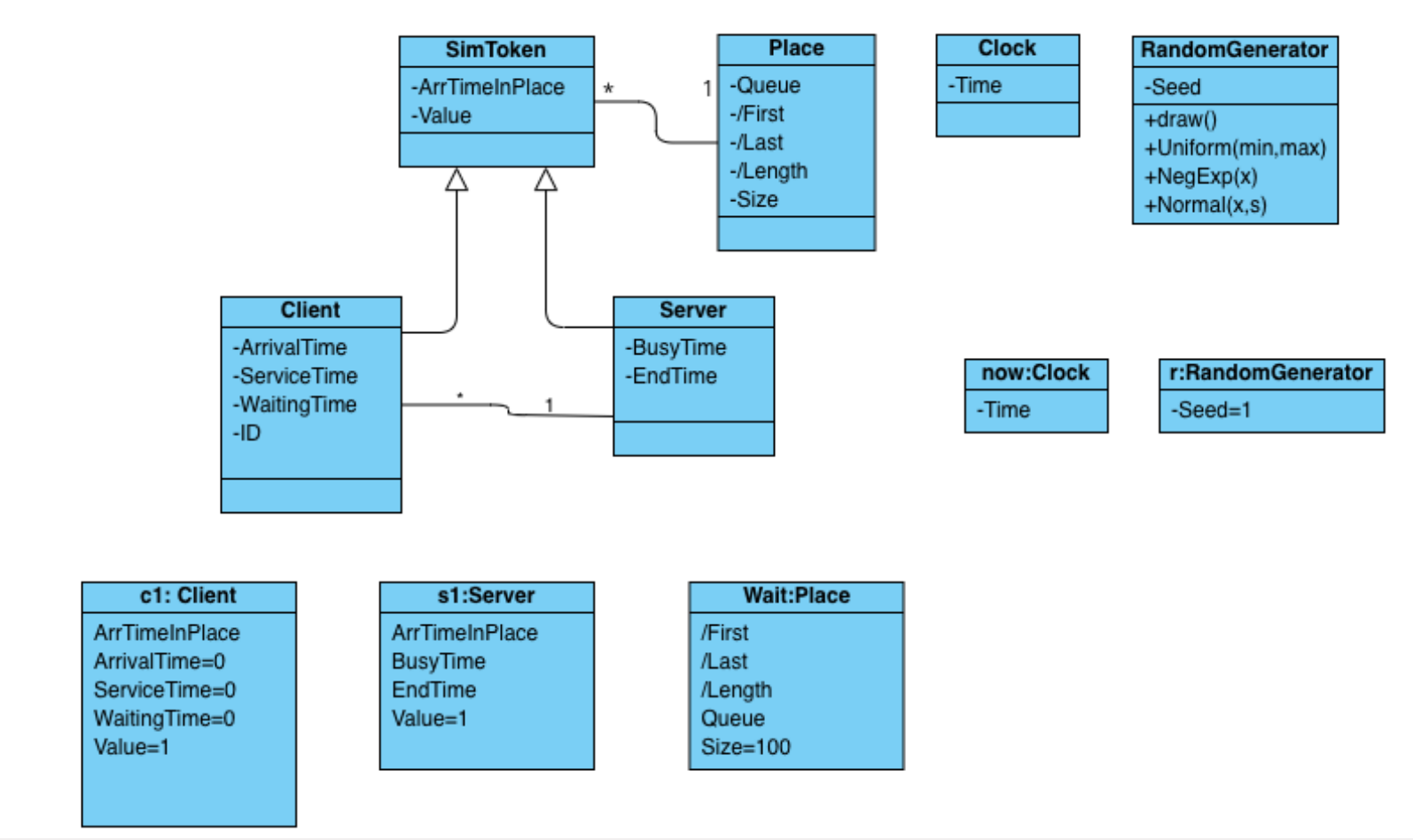
M/M/1 with maximum of 100 customers



Arrive
Pre c.ArrivalTime = now.Time ∧ **c.ID < 101**
Post Wait := Wait.add(c) ∧
c.ServiceTime := 0 ∧
c.WaitingTime := 0 ∧
New :=New.add(n) ∧
n.ArrivalTime := c.ArrivalTime + r.NegExp(5) ∧
n.ID := c.ID + 1 ;

Return
Pre **c in Served**
Post **Wait := Wait.add(c) ∧**
c.ArrivalTime := c.EndTime + r.NegExp(5)

EXAMPLE: closed system



Summary of today's lecture

STEP 3: Conceptual model

- Process model (classical petri net)
- Information model (UML diagram)
- Specifications (initial situation, transitions, measurement functions)

Examples of modeling patterns and design decisions

- Generating clients
- Maximum queue length + early departure of customers
- Switching queues
- Queue discipline (FIFO/SPT)

Manual simulation

Homework

For next lecture:

- Install the SimPN library at https://github.com/bpogroup/simpn