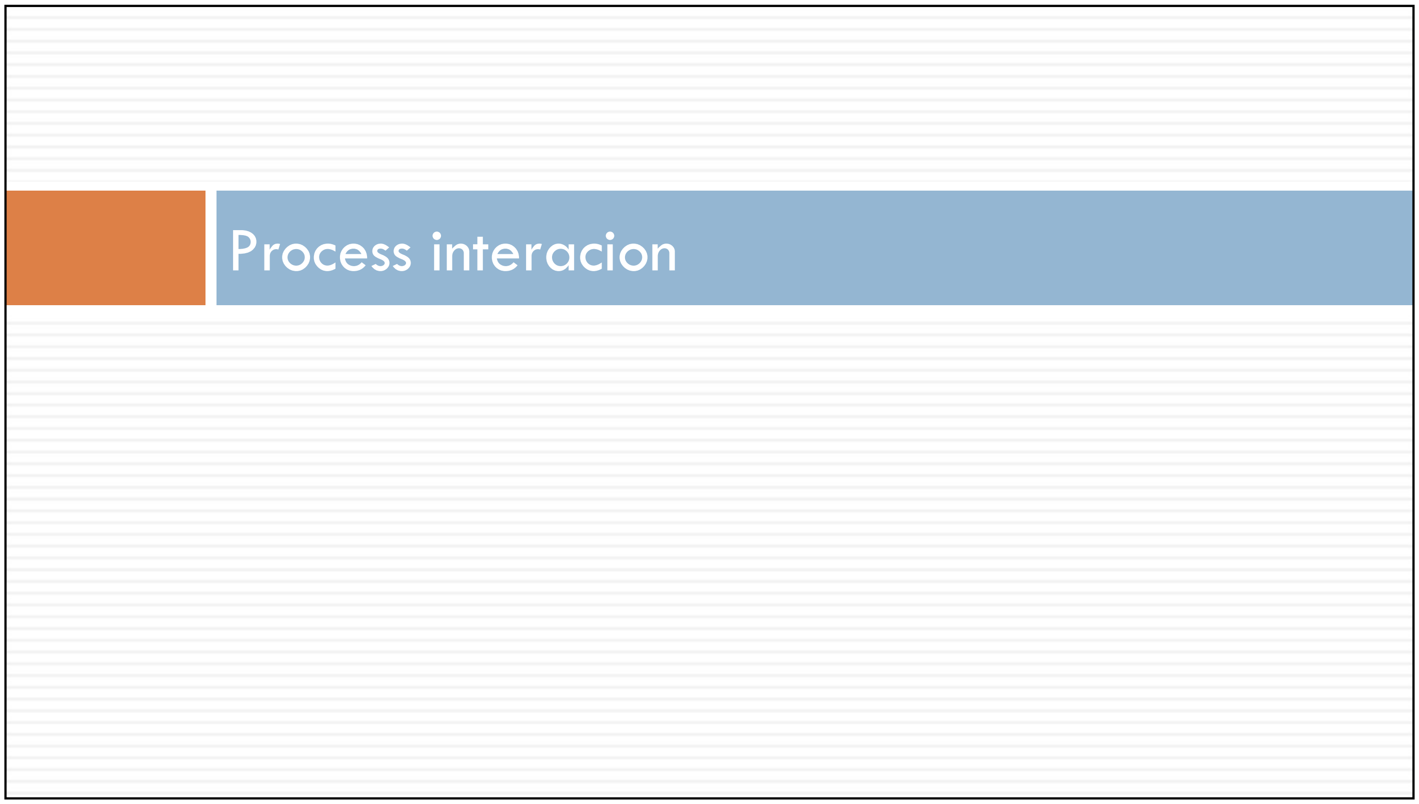
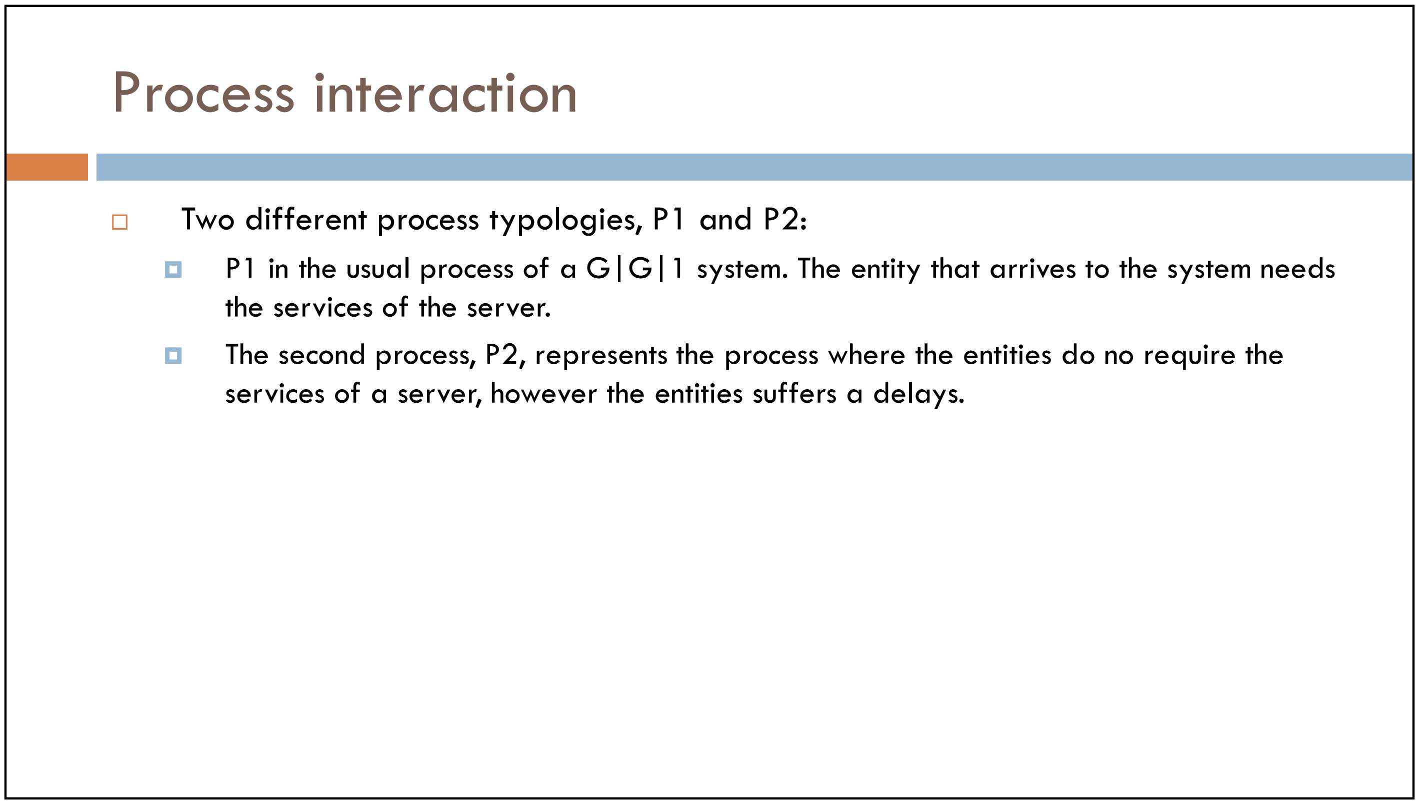


Process interacion

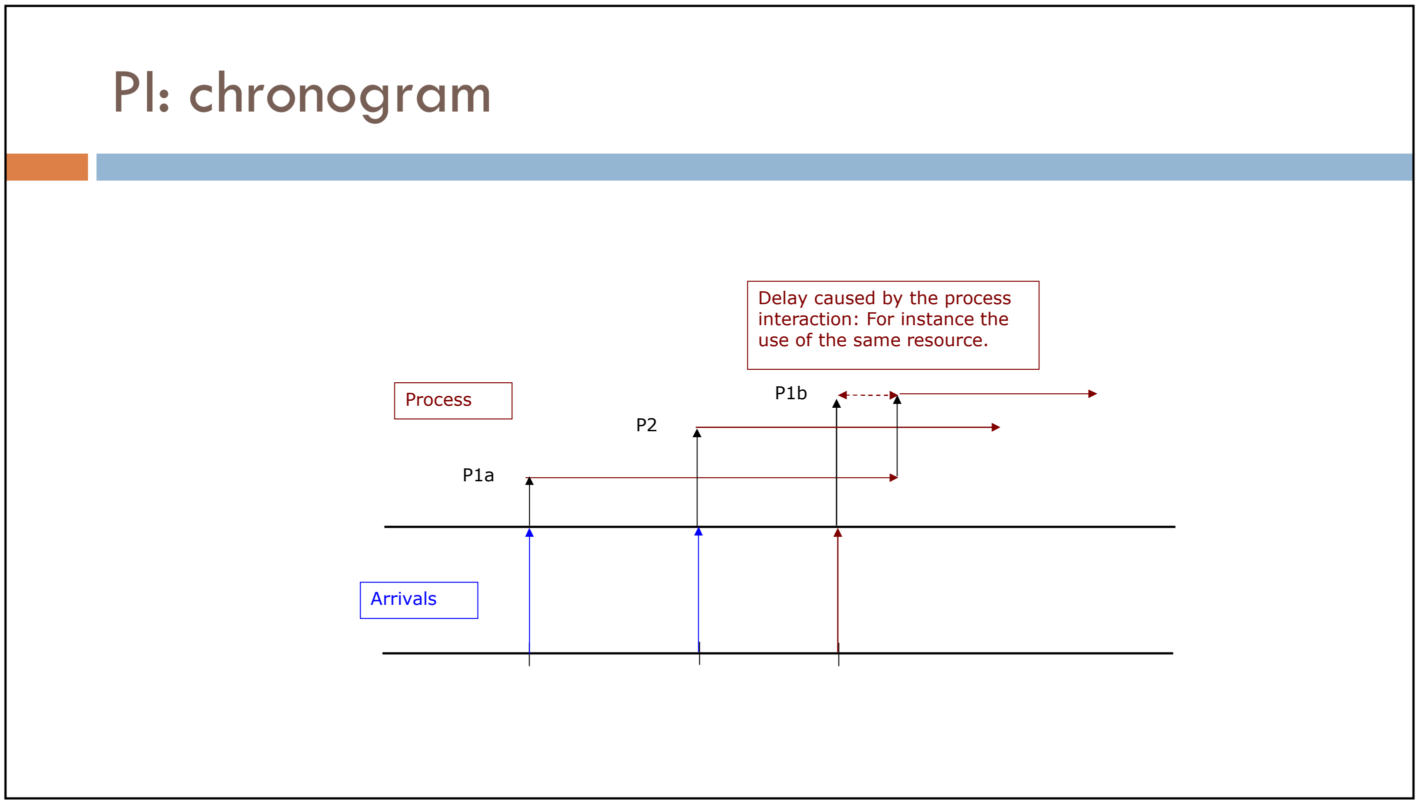
The last method to create a simulation engine, is Process interaction. In this approach we will need to define the semantics of the elements that will be used to represent the process, the behavior that rules the movement of our entities. We will use GPSS as a language to understand this semantics.



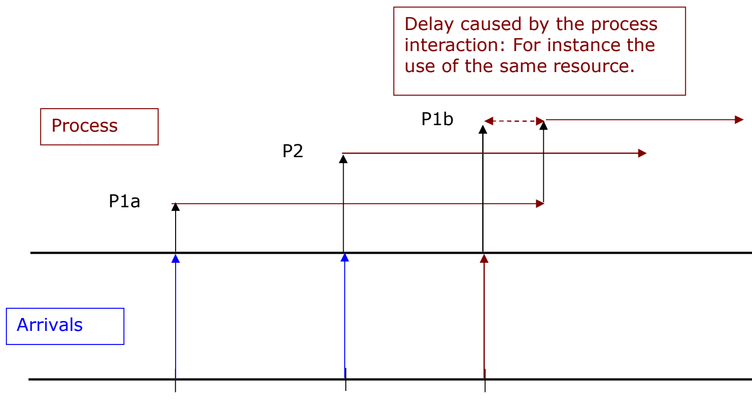
Process interaction

- Two different process typologies, P1 and P2:
  - P1 in the usual process of a G|G|1 system. The entity that arrives to the system needs the services of the server.
  - The second process, P2, represents the process where the entities do no require the services of a server, however the entities suffers a delays.

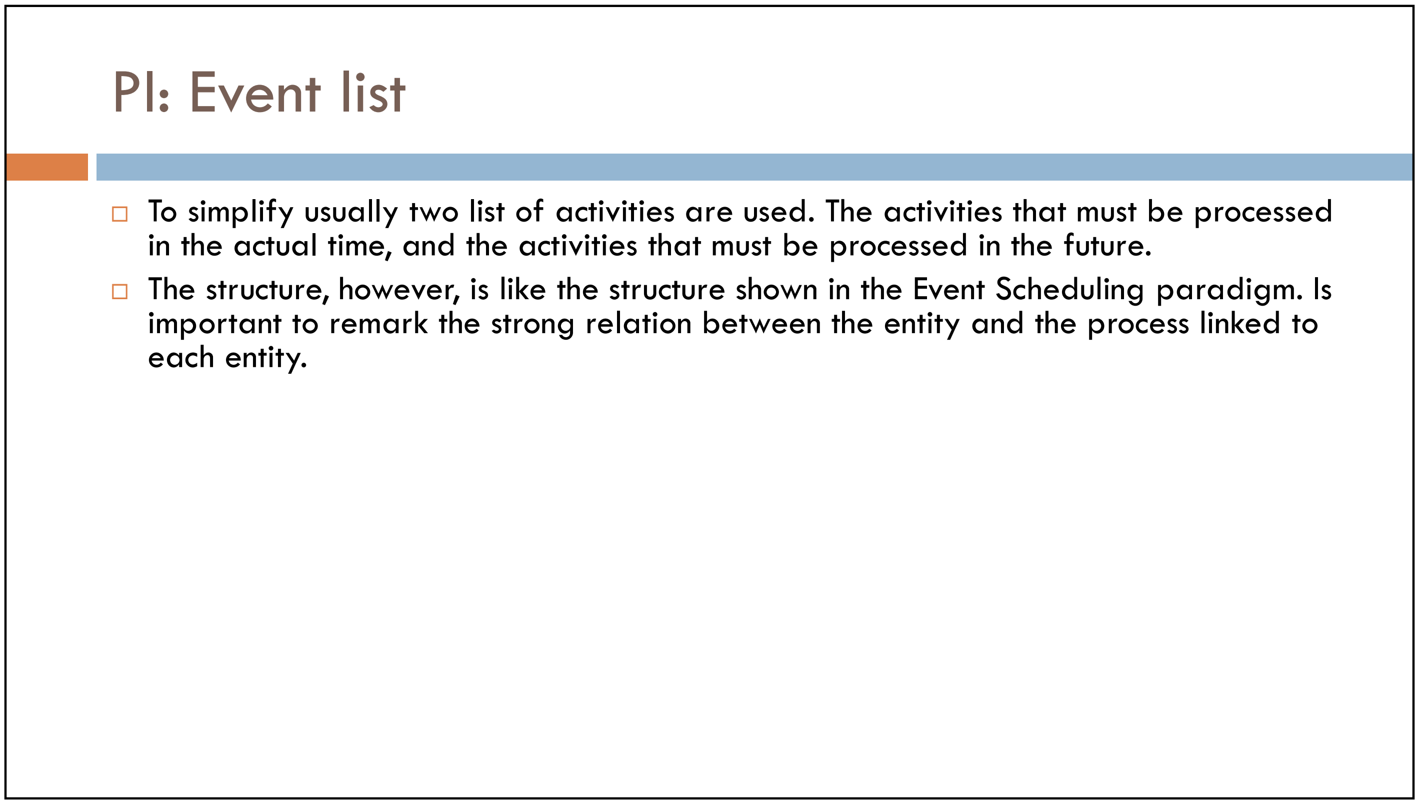
Conceptually we consider that we have two typologies for the processes, P1 process that are those processes that needs a server to complete the different actions, and P2 pocess, that are those process that do not need a server, but needs some time to complete the actions. This is a conceptual approach to the process we will use in simulation based on Process Interaction engine, since no pure P1 or P2 processes exists.



PI: chronogram


Delay caused by the process interaction: For instance the use of the same resource.
Process
P1b
P2
P1a
Arrivals

This will be the chronograph we will use to represent the behavior of the model. Notice that P1b process can only start its different actions until P1a process finish, since they uses the same server. Meanwhile a P2 process, that do not need any server, completes its actins without any interference with any other processes.

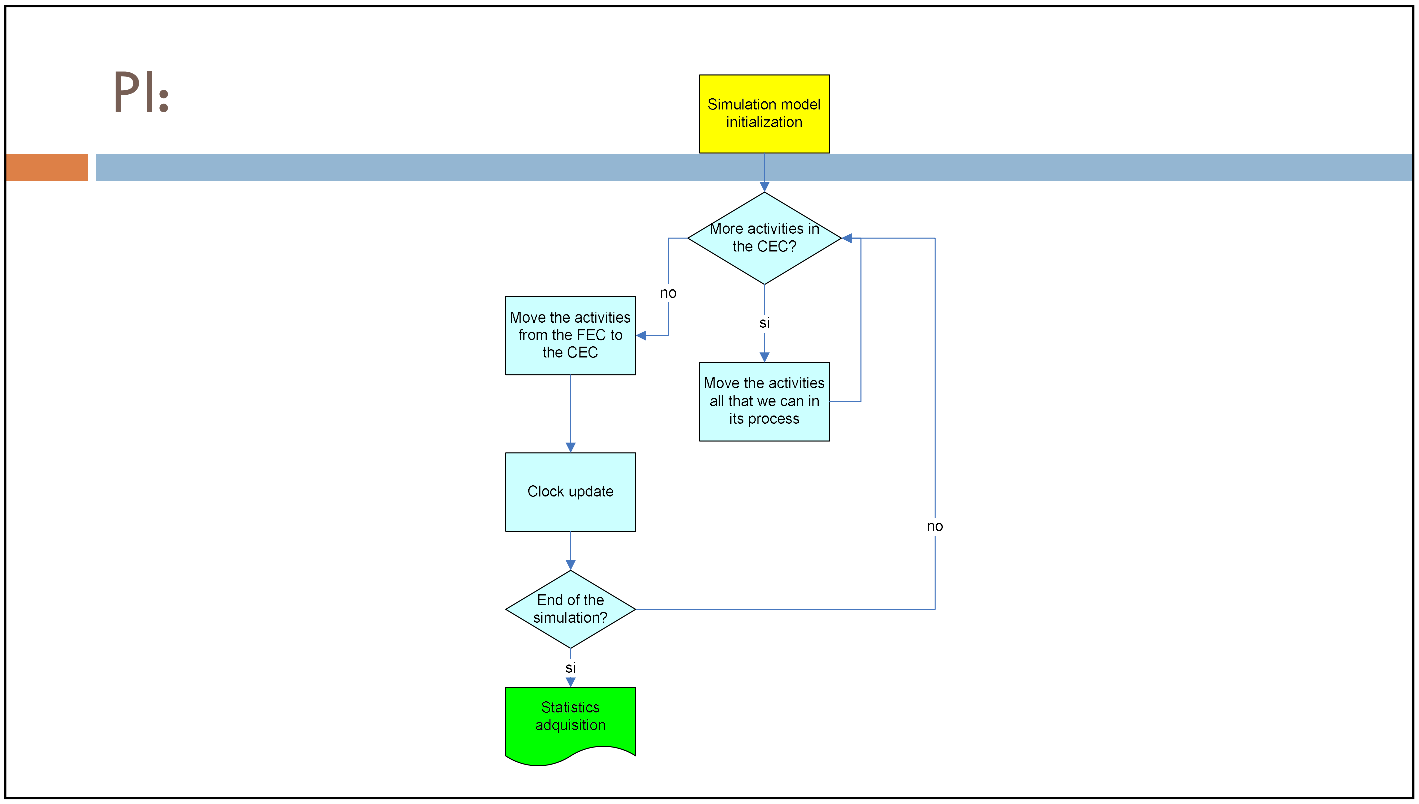


PI: Event list

- To simplify usually two list of activities are used. The activities that must be processed in the actual time, and the activities that must be processed in the future.
- The structure, however, is like the structure shown in the Event Scheduling paradigm. Is important to remark the strong relation between the entity and the process linked to each entity.

The event list will be like the event list in Event Scheduling, but to simplify the interaction of the engine with the event list, we will define two event list, one to process the current events, the Current Event Chain, (CEC), and other list to process the future events, the Future Event Chain, (FEC).

On the CEC the events, that represent an entity in a specific position of its process, will be sorted by priority, while on the FEC, the entities will be sorted by time.

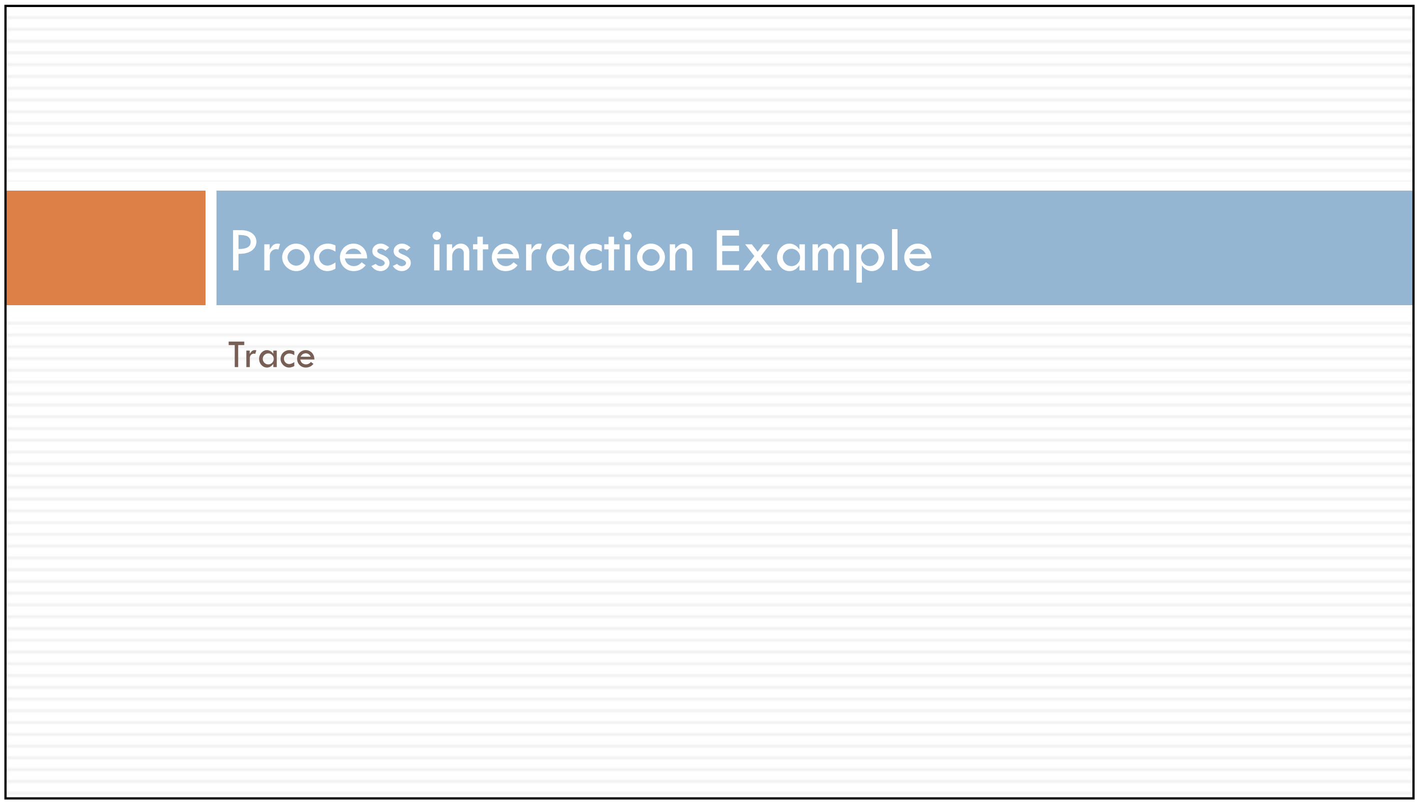


PI:

Simulation model initialization
More activities in the CEC?
no
si
Move the activities from the FEC to the CEC
Move the activities all that we can in its process
Clock update
no
End of the simulation?
si
Statistics adquisition

With this considerations the algorithm that rule the engine behavior is presented on this slide. First, we perform, as usual, the initialization of the model. Later we will analyze if there is any event on the CEC tat needs to be processed. In case that exist an evet, we will move the transaction, the entity, as far as we can on its process. This can be done until this entity leave the model (crosses a TERMINATE in GPSS syntax), leaving the model, finds a delay (ADVANCE in GPSS syntax) , in that case the entity must be send to the FEC, or finds a server that is user by any other entity, implying that this transaction will be stopped on the CEC.

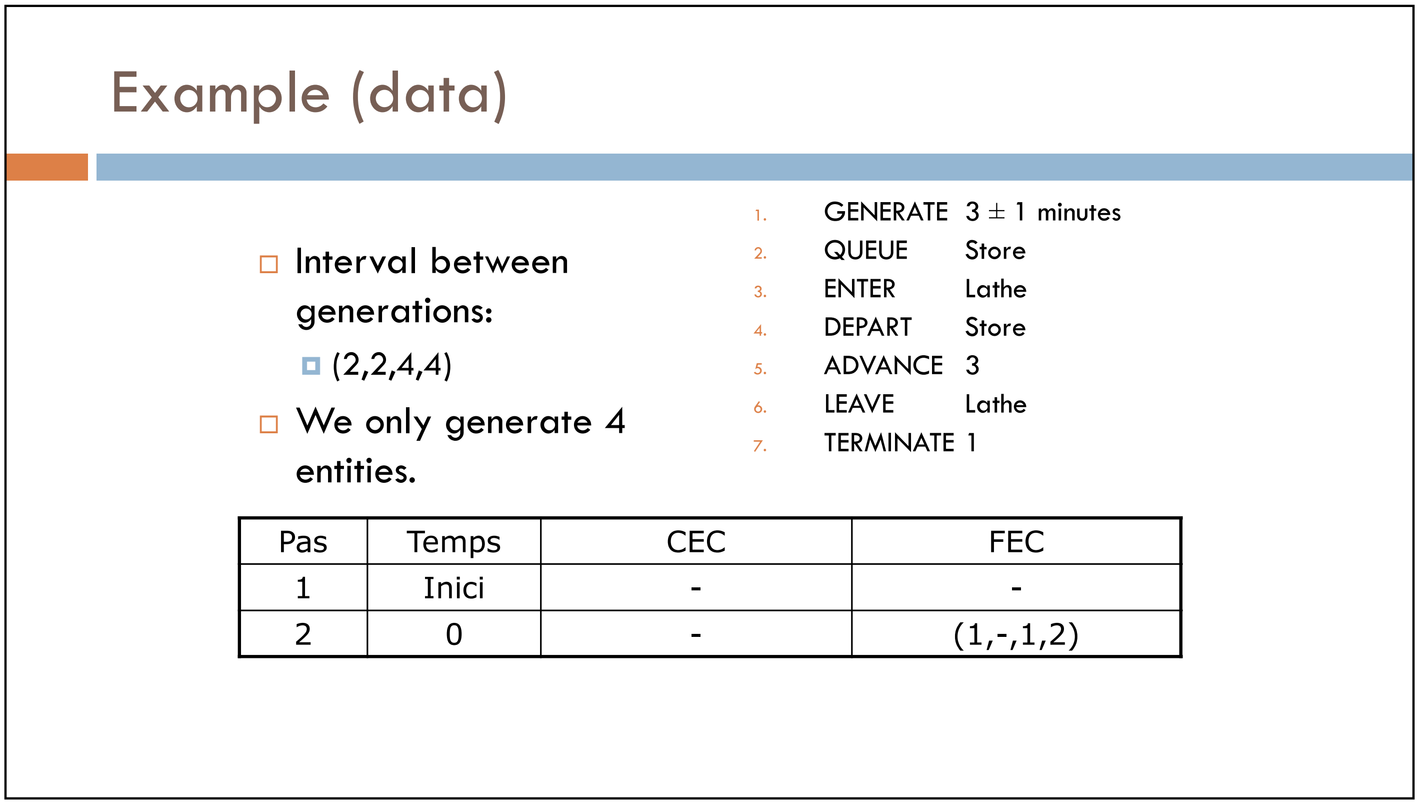
When no more entities can be processed on the CEC, because this list is empty or because the entities are on the CEC blocked in front of a server, we will go to the FEC, and we will move all the entities that owns the smallest time to the CEC. We update the simulation clock with this time. We also analyze if we finish the simulation (often defined by time) if this is the case we write the simulation report, if not, with new entities (events) on the CEC we can start processing again.



Process interaction Example

Trace

We will review now the process interaction approach through an example.



Example (data)

- Interval between generations:
  - (2,2,4,4)
- We only generate 4 entities.

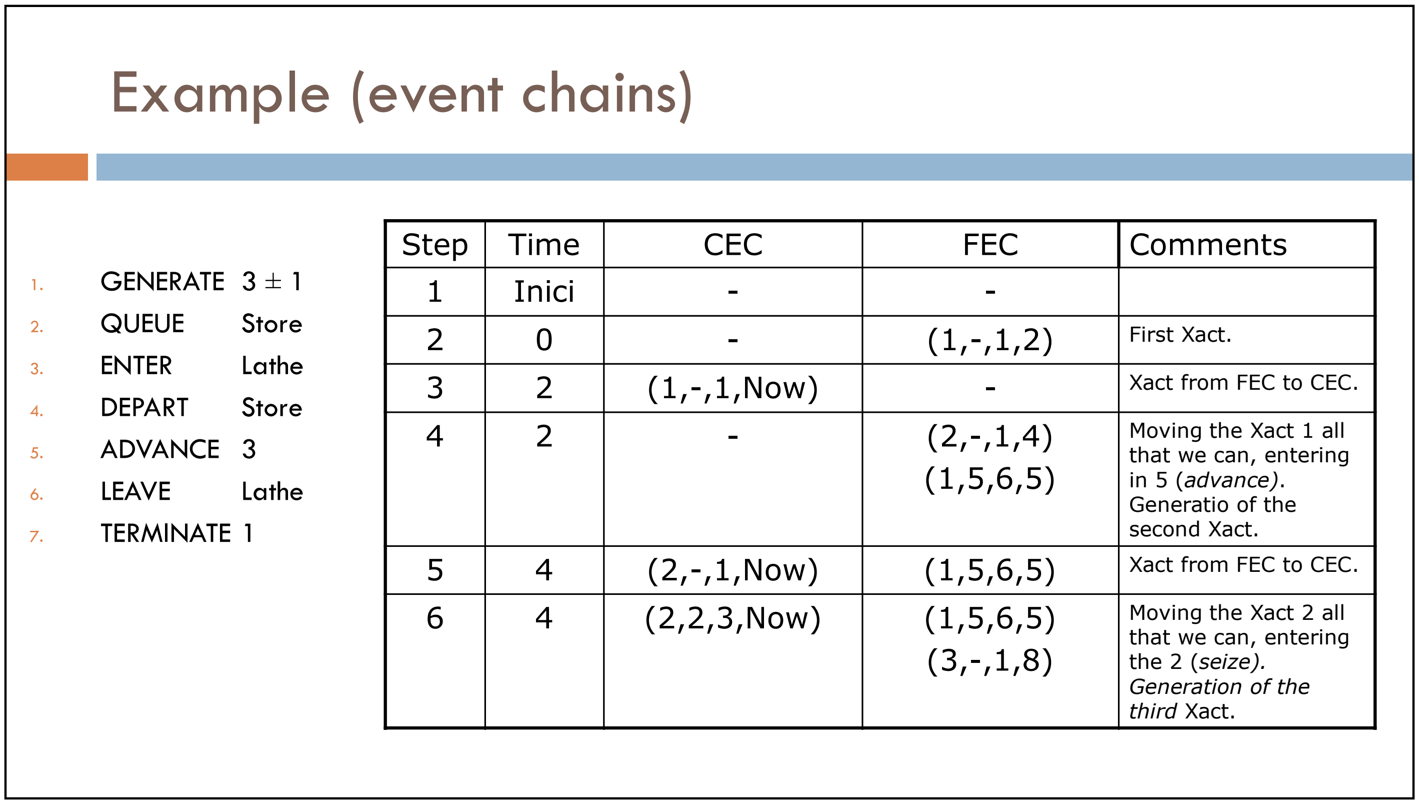
1. GENERATE 3 ± 1 minutes
2. QUEUE Store
3. ENTER Lathe
4. DEPART Store
5. ADVANCE 3
6. LEAVE Lathe
7. TERMINATE 1

Pas	Temps	CEC	FEC
1	Inici	-	-
2	0	-	(1,-,1,2)

This example will represent a simple queue model; notice that we are following the GPSS approach on the semantics of the different elements that compose the PROCESS.

The time between generations is a uniform distribution between 3 plus minus 1 second, and we consider that we have the values generated at the beginning. 2,2, 4 and 4 will be the values we will use to represent the generation between entities.

The table presented below represents the structure of the simulation trace we will use. Notice that the structure inside the table represents, on the first position the identifier of the entity, in that case 1. The second value will represent the current position of the entity. The position represent the number of the operations (GPSS block) that the entity occupies. The minus signs represents that the entity is outside the system. The third element represent where wants to go the entity. In the tuple shown on the table, the entity wants to go to the first position, the GENERATE to be introduced on the model process. Finally, the last element of the tuple represents when this action, the movement from outside the model to the first position, the GENERATE, will take place, on time 2, that is the first intergeneration times.



Example (event chains)

1. GENERATE 3 ± 1
2. QUEUE Store
3. ENTER Lathe
4. DEPART Store
5. ADVANCE 3
6. LEAVE Lathe
7. TERMINATE 1

Step	Time	CEC	FEC	Comments
1	Inici	-	-	
2	0	-	(1,-,1,2)	First Xact.
3	2	(1,-,1,Now)	-	Xact from FEC to CEC.
4	2	-	(2,-,1,4) (1,5,6,5)	Moving the Xact 1 all that we can, entering in 5 (*advance*). Generatio of the second Xact.
5	4	(2,-,1,Now)	(1,5,6,5)	Xact from FEC to CEC.
6	4	(2,2,3,Now)	(1,5,6,5) (3,-,1,8)	Moving the Xact 2 all that we can, entering the 2 (*seize). Generation of the third* Xact.

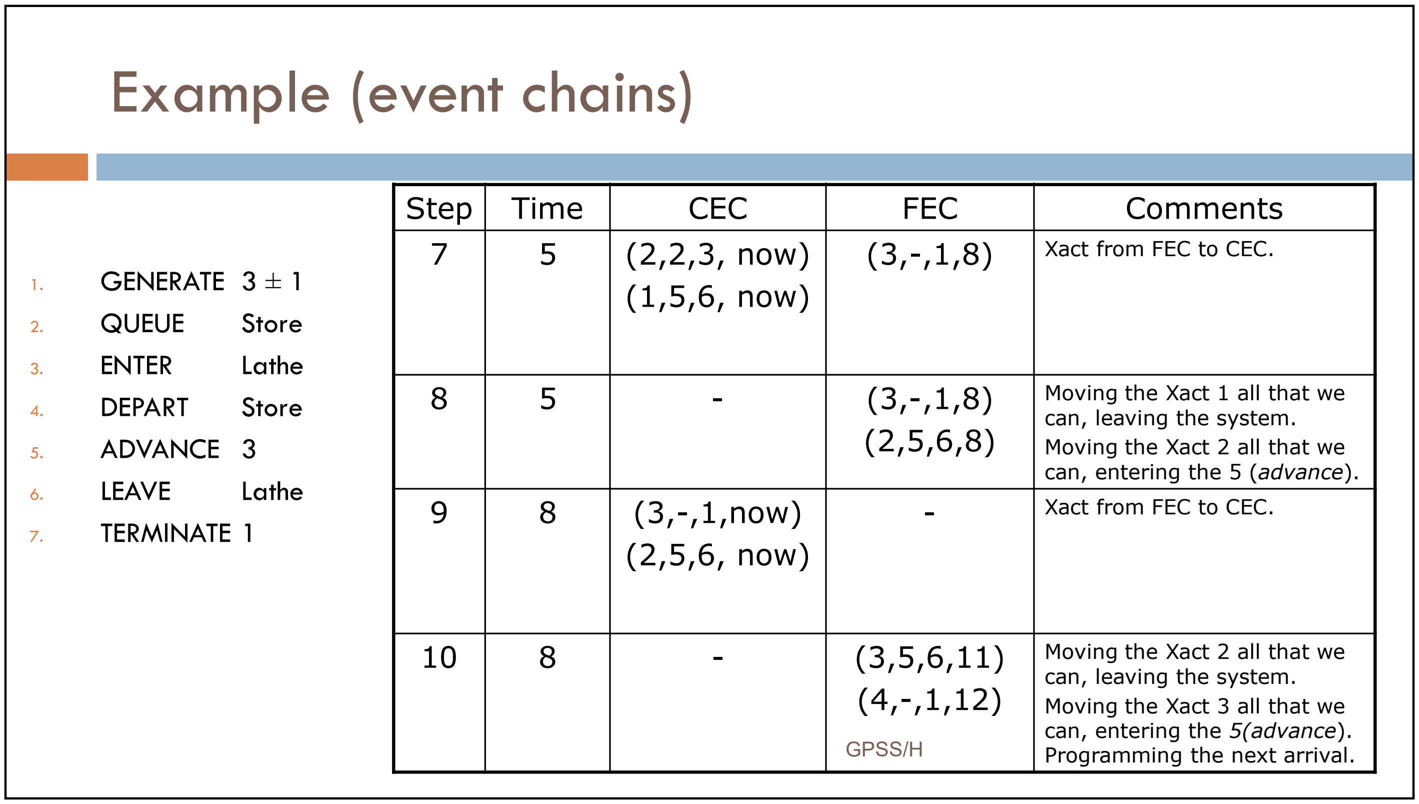
With this and following the process interaction approach we can generate the movement of the entities through the different process actions.

At the beginning the first entity moves from outside to the GENERATE, On the GENERATE the entity is on the FEC, because the time is 0, hence, and because the CEC is empty, we must move all the entities from the FEC to the CEC. This is done in the third steep. On this steep the clock is updated to 2 and the entity crosses the GENERATE.

On steep 4, the first entity, because crosses the GENERATE, generates a new entity, the second, that will be on the FEC, because wants to do the movement from the outside to the first activity on time 4, because the time now is 2 and the second inter generation time is 2. The first transaction continues its movement through the different activities until the ADVANCE, where it must remain for three units of time. Hence the first transaction will be on block 5, the ADVANCE, wants to go to the next block the LEAVE, the 6th block, and wants to do this 3 units of time later, that is the time related to the delay, hence on time 5.

Since at steep 4 nothing more can be done, we see if there are entities on the FEC that can be transferred to the CEC, we detect one, the second entity. This will be transferred to the CEC, see steep 5, and starts its movement, steep 6. Since on steep six, the second entity also crosses the GENERATE, a new entity is generated. This new entity, the third will remain outside of the model until time 8, because the time between generation for the fourth entity is 4, and the current time is 4.

The second transaction starts its movement on the process until it reaches the second block, since the server is used by the first entity, hence this second entity will remain on the queue, until the first entity leaves the server.



Example (event chains)

1. GENERATE 3 ± 1
2. QUEUE Store
3. ENTER Lathe
4. DEPART Store
5. ADVANCE 3
6. LEAVE Lathe
7. TERMINATE 1

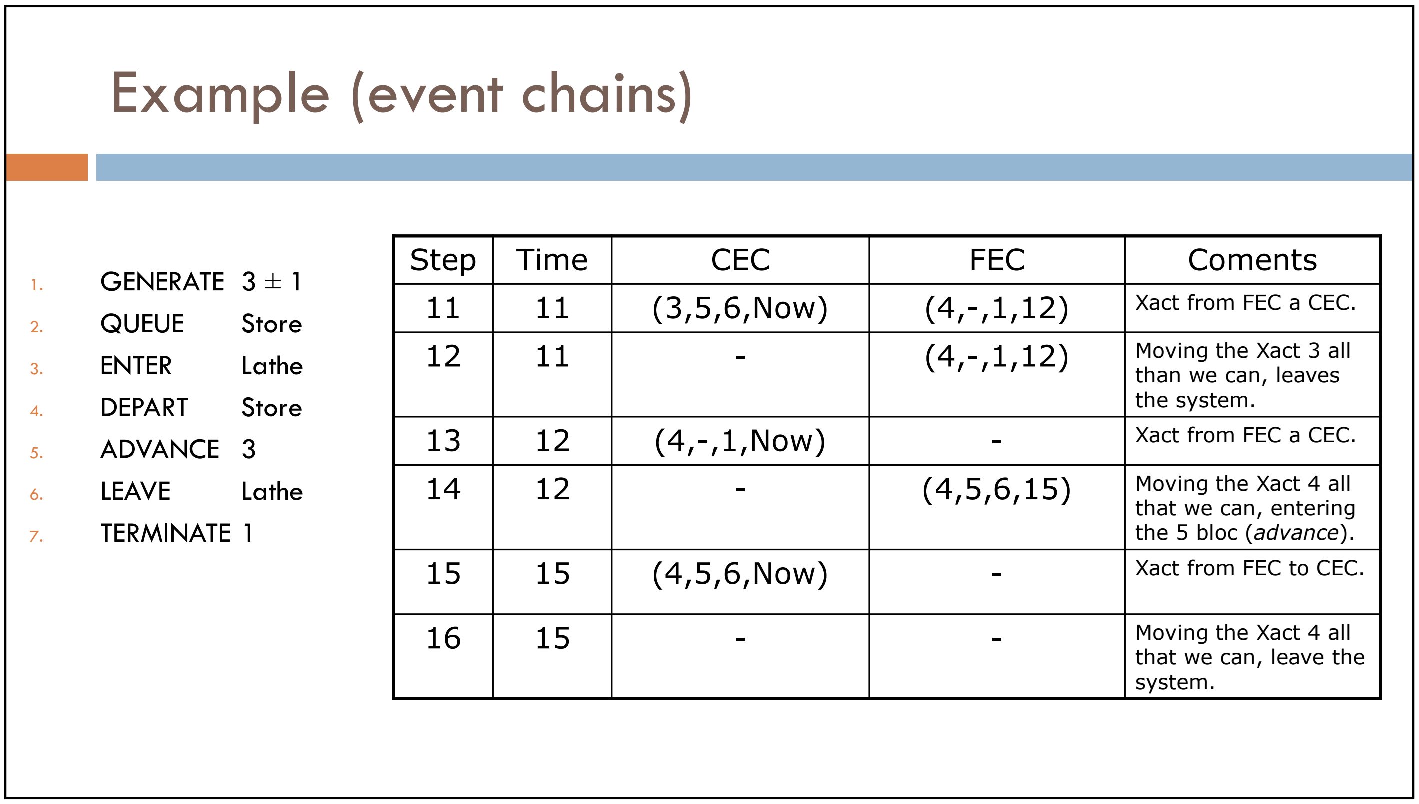
Step	Time	CEC	FEC	Comments
7	5	(2,2,3, now) (1,5,6, now)	(3,-,1,8)	Xact from FEC to CEC.
8	5	-	(3,-,1,8) (2,5,6,8)	Moving the Xact 1 all that we can, leaving the system. Moving the Xact 2 all that we can, entering the 5 (*advance*).
9	8	(3,-,1,now) (2,5,6, now)	-	Xact from FEC to CEC.
10	8	-	(3,5,6,11) (4,-,1,12) GPSS/H	Moving the Xact 2 all that we can, leaving the system. Moving the Xact 3 all that we can, entering the *5(advance*). Programming the next arrival.

On steep 7 the entitles are transferred from the FEC to the CEC and the time is updated to 5. Now entity 1 can continue its movement and freely leaves the system.

On steep 8, the entity 2 is going to start its movement through the server, and will be stopped by the ADVANCE, that will retain it until time 8, that is 5 plus 3.

On the steep 9 the entity 3 enters on the process, and crosses the GENERATE, introducing the entity 4, also the entity 2 continues its movement and leaves the system.

On steep 10, the entity 3 is stopped on the ADVANCE until time 11, and the last entity, identified by 4 will wait to enter in the process at time 12.



Example (event chains)

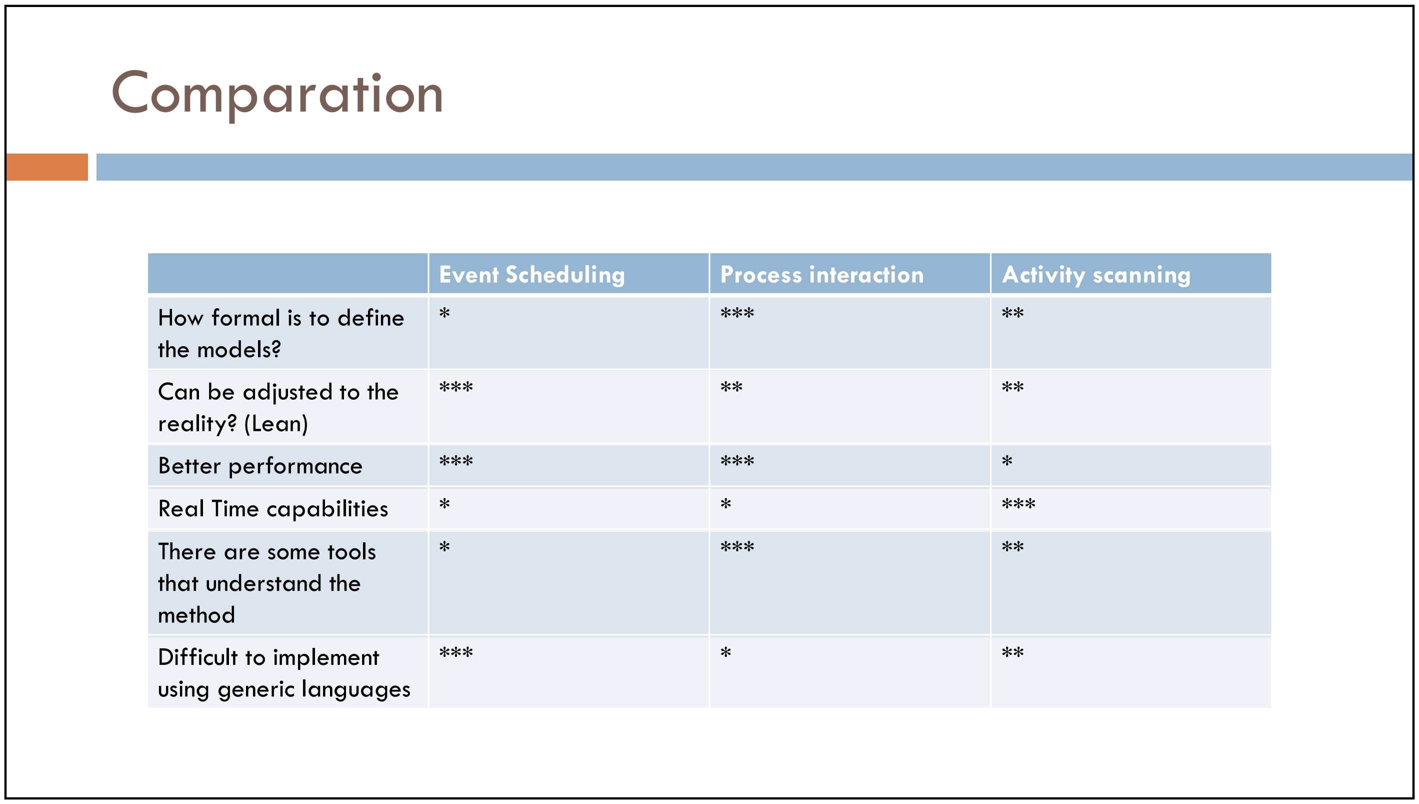
1. GENERATE 3 ± 1
2. QUEUE Store
3. ENTER Lathe
4. DEPART Store
5. ADVANCE 3
6. LEAVE Lathe
7. TERMINATE 1

Step	Time	CEC	FEC	Coments
11	11	(3,5,6,Now)	(4,-,1,12)	Xact from FEC a CEC.
12	11	-	(4,-,1,12)	Moving the Xact 3 all than we can, leaves the system.
13	12	(4,-,1,Now)	-	Xact from FEC a CEC.
14	12	-	(4,5,6,15)	Moving the Xact 4 all that we can, entering the 5 bloc (*advance*).
15	15	(4,5,6,Now)	-	Xact from FEC to CEC.
16	15	-	-	Moving the Xact 4 all that we can, leave the system.

On steep 11 entity 3 leaves the System.

On steep 12, 13, 14, 15 and 16 we can see that the last entity freely moves on the process, because on other entity exists and it finds the sever free. The only stop is due the time needed to do the action on the ADVANCE, on steep 14.

Now the system don’t have more entitles and the simulation finish.



Comparation

	Event Scheduling	Process interaction	Activity scanning
How formal is to define the models?	*	***	**
Can be adjusted to the reality? (Lean)	***	**	**
Better performance	***	***	*
Real Time capabilities	*	*	***
There are some tools that understand the method	*	***	**
Difficult to implement using generic languages	***	*	**

Here you have a comparison between the different simulation engines. All of them have some particularities that makes the interesting depending on what we prefer, more stars implies that it will perform better.