DISCRETE SIMULATION ENGINES

Pau Fonseca i Casas; pau@fib.upc.edu

On this slides we will discuss regarding Simulation engines, that can be used to execute discrete simulation models.

Discrete simulation engines

- Event Scheduling
  - Programació d'esdeveniments
  - Programación de eventos.
- Process interaction
  - Interacció de processos
  - Interacción de procesos.
- Activity scanning
  - Exploració d'activitats
  - Exploración de actividades.

Mainly there are three ways to execute a simulation model, Event Scheduling, Process interaction and Activity scanning. Historically the first method that appears was Activity Scanning, latter Event Scheduling and finally Process Interaction.

Event Scehduling

Let ’s start with Event Scheduling that is the one that can serve as a basis to understand faster the other two approaches. The system that we will use as an example will be a simple queue model composed by a single server. Notice however, that in simulation we do not have the constrains that exist in queuing theory.

ES: example

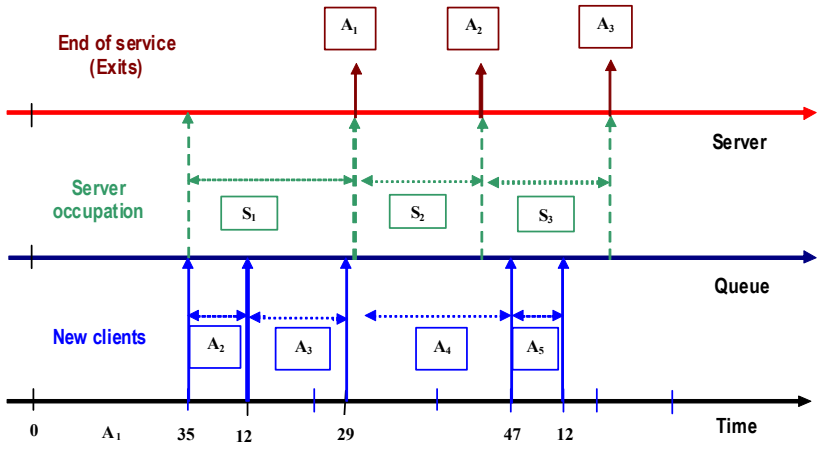
Time between arrivals		Service time	
a_1	35	b_1	40
a_2	12	b_2	30
a_3	29	b_3	30
a_4	47	b_4	20
a_5	12	b_5	30

On this table are represented the different events that rules the modification of the system state variables. Since the model is a queue system, we only have a queue and a server, hence we will detail only two event that can cause that the state of the system changes (the number of elements in the system). These two events are the arrival and departure event.

For the shake of simplicity we will detail the values of this events on this lists but remember that these values will be generated through a Random Variable Generator that expresses how this phenomenon takes place.

ES: chronogram

Time between arrivals		Service time	
a_1	35	b_1	40
a_2	12	b_2	30
a_3	29	b_3	30
a_4	47	b_4	20
a_5	12	b_5	30

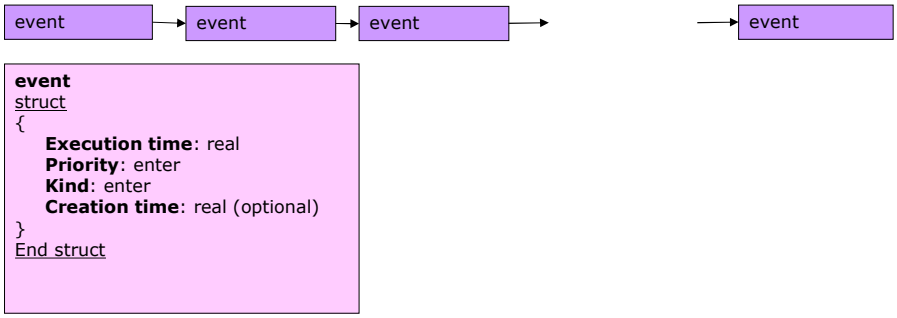


Once we have this values, we can define a chronogram that express the arrivals and the departures of the elements through the system, understanding the accumulations that can appear. With this chronograph we will be able to calculate the queues and the accumulations on the system.

The first arrival takes place at time 35, since the system is empty, this first entity, takes the server and starts the service. This last for 40 units of time, and then leaves the system. Notice that during this first service, two other entities enters in the system, increasing the number of elements in the queue until 2. When the first entity leaves the system, the second captures the server and start this second service, that last for 30 units of time. When this second entity leaves the system, the third entity enters on the server and starts its process. At this time the number of elements on the queue is again 0.

Notice that with this chronogram and following this approach we are able to calculate the statistical information we need for the analysis of this type of systems. Hence Event Scheduling approach will be focused on the definition of this type of chronograph,

ES: Event list



The EVENT is the main element in discrete Simulation, and special relevant in Event Scheduling approach. The structure is shown on this slide, where the tuple have:

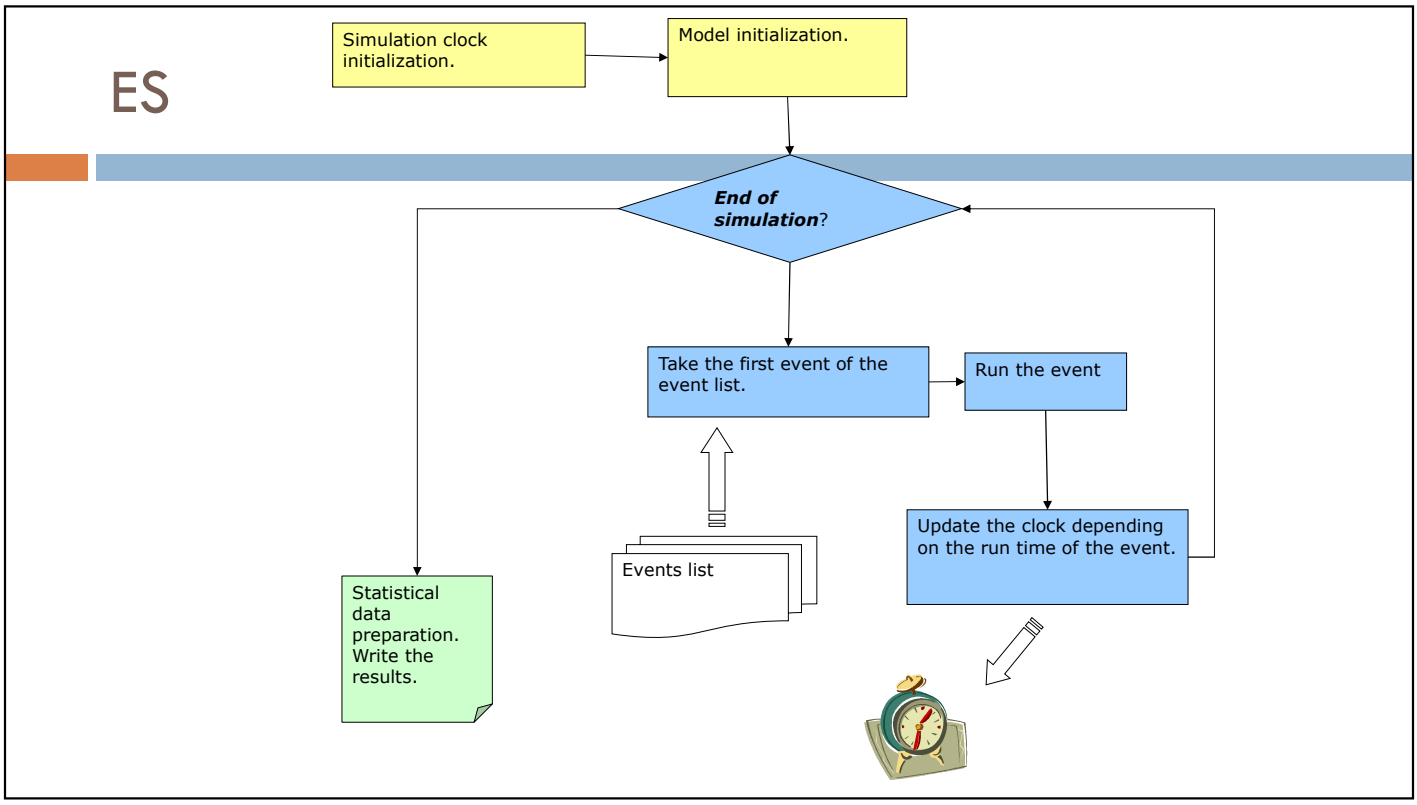
Execution time, or event time, that shows when the simulation engine must run the event.

Priority, used to define what to process first if two events arrives with the same execution time.

Kind of event, a value to identify the nature of the event, in the example if we are analyzing an arrival or a departure, depends on the model definition.

Creation time, that shows the time when the event enters in the simulation system. This is optional and used only to obtain statistical information.

The events are sorted in a structure similar to an event list. In the commercial tools this is not an event list, since we want a structure that is faster to allow adding and deleting elements, the events. The event list is sorted by time and later by priority, this assures that at the beginning of the list we can find always the event that must be processed.



With all this information, now we can detail the Event Scheduling algorithm that rules the execution of a model.

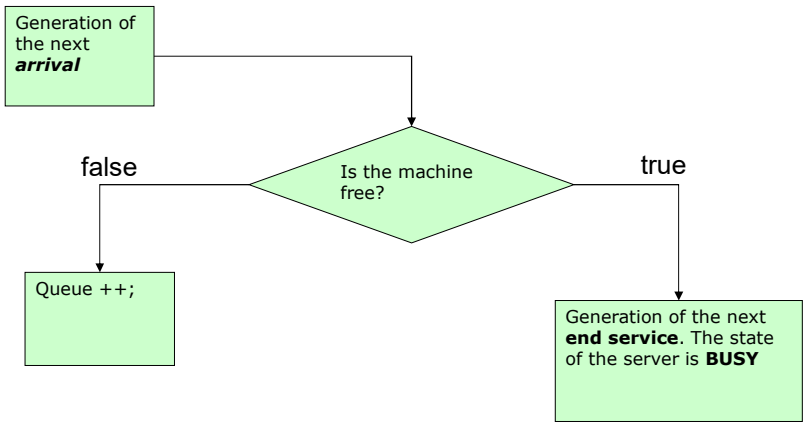
First, we do the initializations to the simulation clock, usually this will be put the global value to zero, and do the initialization of the model, that also usually represents to start with an empty model, no elements on the system. With this elements we will define the initial conditions of the simulation model.

Once to this initialization process, we analyze if the simulation finish condition is meet, often this is a temporal condition, as an example, we conduct a simulation for one year, however other conditions can be used to finish the simulation, like the number of entities that leave the system, or the number of operations done, among others.

If the simulation is not finished, we will go to the event list ant we will take the first event. Depending on the type of the event we will execute one or other procedure, and because the event have an execution time related, we will update the simulation clock accord to this.

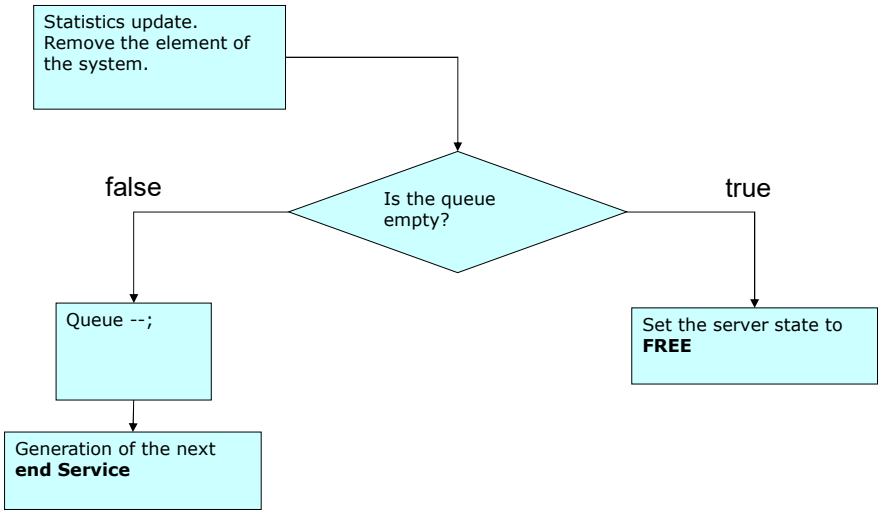
On the other hand, if the simulation ends, we will recover the statistical information and we will write the results.

ES: Arrive event procedure



Notice that is relevant to describe the procedures that rules the behavior of each event we have. In that sense, the schema that expresses the arrive procedure is detailed on this slide. Notice that the first thing we do is to generate a new event, without this the simulation will finish because the lack of arrivals. Once we have this arrival we will detect if the machine, the server we have, is free or not. If the machine is free, we will change the state of the server to BUSY and we will generate the finish service time. If not, we will increase the number of elements in the queue.

ES: Exit event procedure



For the exit event we detail here the procedure that rules its behavior. Since an entity leaves the system, the first thing we do is to acquire the statistical information and to remove the element from the simulator, to save memory. Then we see if the queue is empty or not, depending on this we will ser the state of the server to free, If no entities are waiting on the queue, or we take the first element of the queue. In this example, since all the elements of the queue are equal, we can model the queue just with a counter. Taking the first element is just decreasing the queue by one and generating the time needed to do the next End Service.

Event Scheduling Example

Trace

At this point we understand the algorithm that will be used to represent a simulation model; hence we can represent the chronograph that allows to obtain the statistical information from this model execution. This chronograph will be represented by a trace, that we will detail next.

ES: Events evolution

Id	Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
0	0	-	-	0	0	0	0

The simulation trace is just a table that contains the main elements to be considered during the simulation execution. The first element, the Id is just an auto numeric value to identify the different rows. The Time is representing the Execution time of the event; hence it represents the simulation clock. Next arrival column shows the Execution time of the next arrival event if this exists. Next exit is representing the Execution Time of the departure process, again, if the event exists. Server state represents the state of the server, just 1 if the server is used, 0 if is free.
Queue shows the number of elements that we have waiting in the queue, is just a natural number. Finally we have two columns that will represent if in each row we are analyzing an arrival or a departure event.

ES: Events evolution

Id	Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
0	0	-	-	0	0	0	0
	0	0,1509	-	0	0	0	0

At this point we are generating the first arrival, notice that without this first generation, the simulation will never start. The time for this first arrival is 0.1509, and at the initial time, that is defined as 0, there are no other entities in the system.

ES: Events evolution

Id	Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
0	0	-	-	0	0	0	0
	0	0,1509	-	0	0	0	0
1	0,1509	0,5778	0,93940	1	0	1	0

Since the event list only contains a single event, we take it from the event list, and we process it. Now the Time is updated, and we generate the next arrival, that it a time 0.57 and, since the server is free, we can generate the next exit, at time 0.93. Notice that at the column Arrive, we add a 1 value, showing that we are processing an arrival.

ES: Events evolution

Id	Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
0	0	-	-	0	0	0	0
	0	0,1509	-	0	0	0	0
1	0,1509	0,5778	0,93940	1	0	1	0
2	0,5778	1,4772	0,93940	1	1	1	0

Event list

Now we analyze the events of the event list, and we detect that “next arrival” is the event that have the lower Execution time, hence will be the event we will process first. Again we generate a new Next Arrival event, at time 1.47, and, since now the state of the server is 1, is used by the previous entity, we increase the Queue elements by one. Notice also that in column Arrive we note that we are processing again an arrival.

ES: Events evolution

Id	Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
0	0	-	-	0	0	0	0
	0	0,1509	-	0	0	0	0
1	0,15099	0,5778	0,9394	1	0	1	0
2	0,57788	1,4772	0,9394	1	1	1	0
3	0,93940	1,4772	3,5225	1	0	0	1

In this third iteration we process a departure, notice that in the event list is the event with less Execution time. At this point we see if there are elements on the queue, that is the case, hence we decrease the Queue variable by one, and we generate a new Next exit event to represent the departure of this element of the queue that now is being served.

ES: Events evolution

Id	Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
0	0	-	-	0	0	0	0
	0	0,1509	-	0	0	0	0
1	0,1509	0,5778	0,9394	1	0	1	0
2	0,5778	1,4772	0,9394	1	1	1	0
3	0,9394	1,4772	3,5225	1	0	0	1
4	1,4772	1,5657	3,5225	1	1	1	0

On this new steep we analyze again the event list and we detect that the Next arrival event, is again the event with the small Execution time, hence we execute this event. We update the clock with its Execution time, and since the server is sued by a previous entity, we increase the number of elements in the queue by one.

One can continue this simulation until the finish condition is reached, remember that often defined by time, but not only.

Activity Scanning

Now we will review Activity scanning, that have some similarities with Event Scheduling but presents one interesting feature that, in some scenarios makes it the more interesting approach.

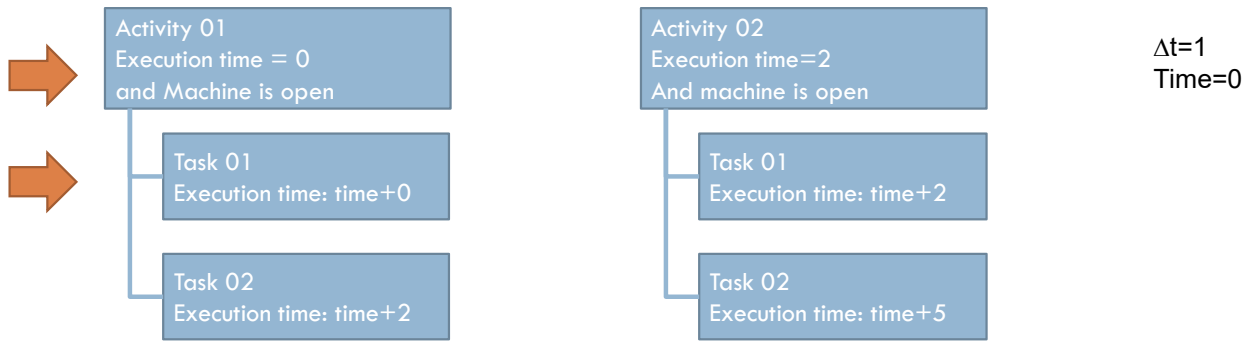
Activity scanning

- Analyze if the simulation engine can run some activity, this depends on the conditions of each activity, and run it until Δt.
- When the simulation engine cannot run more activities increment the clock with Δt.

The more important element in Activity scanning approach is the definition of a delta t, that expresses how the time is going to evolve. This approach differs from Event Scheduling because now the events are going to be processed, analyzing if the simulation clock, that is updated by this delta t reaches the Execution time of the event.

The models will be composed by a set of activities that to be executed depends on several conditions, time, or more usual, a specific state in some elements in a factory, as an example, the state of a machine.

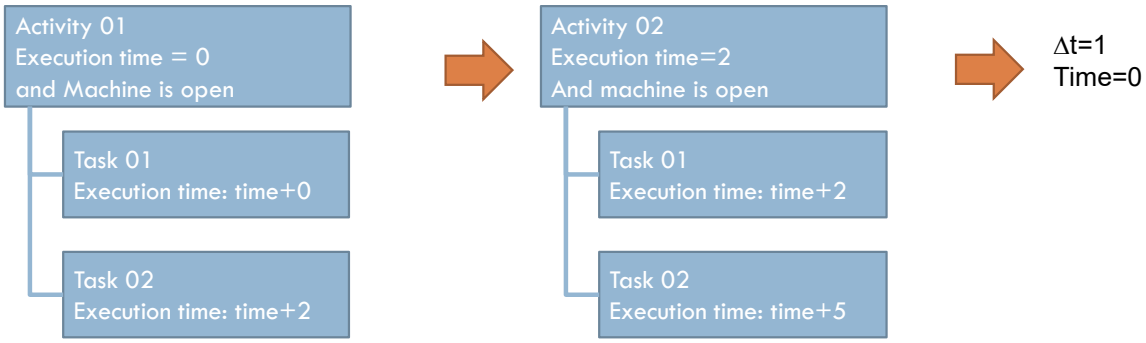
Activity scanning example



On this example we have two activities that are composed by only two tasks. The condition to execute the activities is that there are at least one task that can be executed, because the Execution Time, and that the Machine, a hypothetical element on the system, is open. Notice that this second condition is always true for the shake of simplicity on this example.

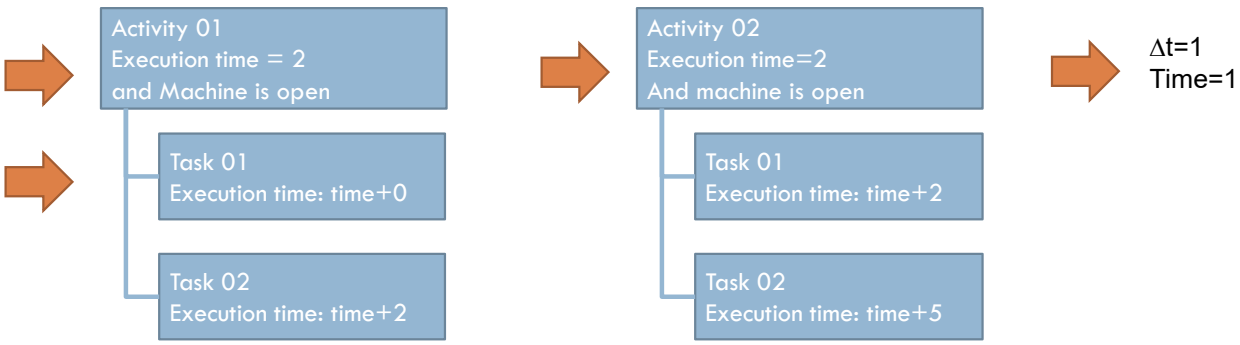
We start with time 0 and we see that the first activity, Activity 01, can be executed, that first task, Task 01 of this activity can be executed, since the time needed to complete the task, from the point of view of the simulation model, not the real time to execute the task, is zero. The second task can be completed at current time plus 2, thence at time 2.

Activity scanning example



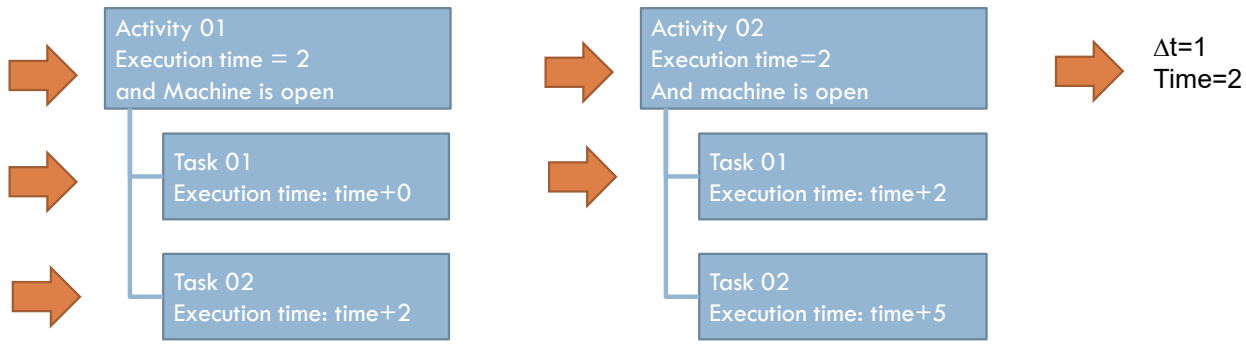
On the second Activity, we try to execute the first task but we cannot do that, hence nothing more can be don on this time, hence we will increase the time using the delta t.

Activity scanning example



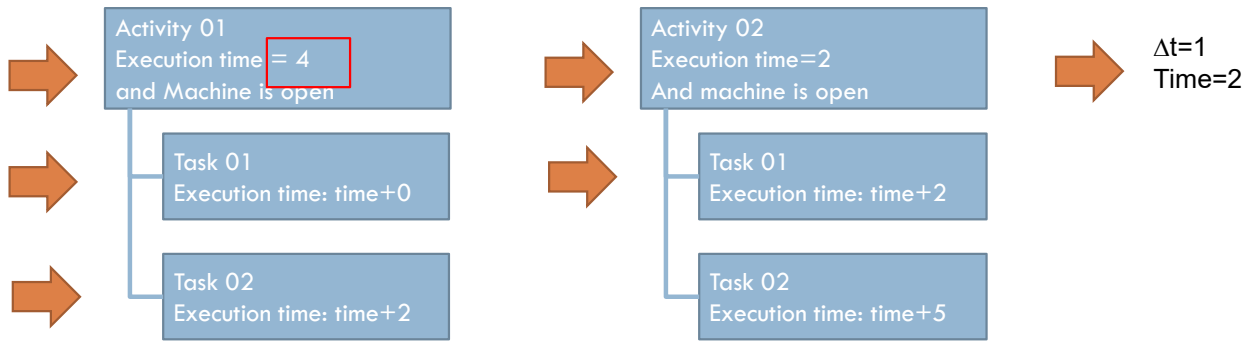
We review all the guards on all the Activities and we detect that nothing can be executed at time one, hence we will increase the time by delta t, one in that case, to time 2.

Activity scanning example



At time two we detect that now we can execute Task 02 from the first Activity and we finish all the Task of this activity, we will start again with this Activity to see if we can execute again more tasks, As we can see, Activity 01 can be executed again.

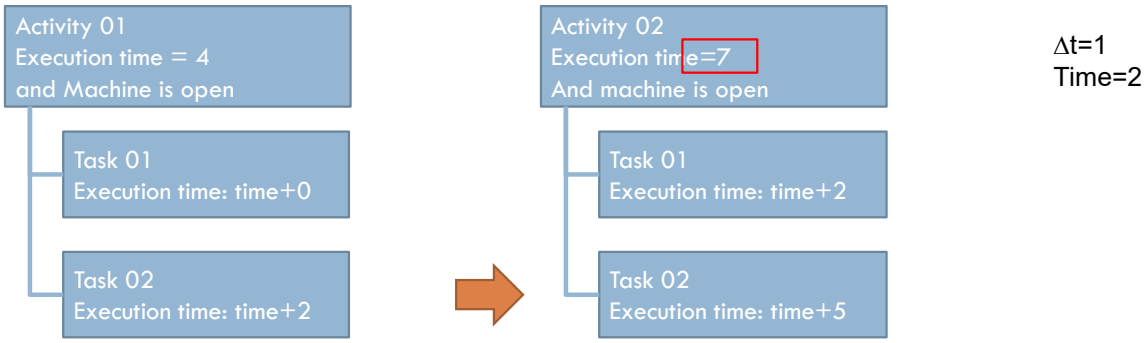
Activity scanning example



Hence, we execute again until Task 02, that needs two more units of time, we express this changing the Execution time constrain in the Activity 01.

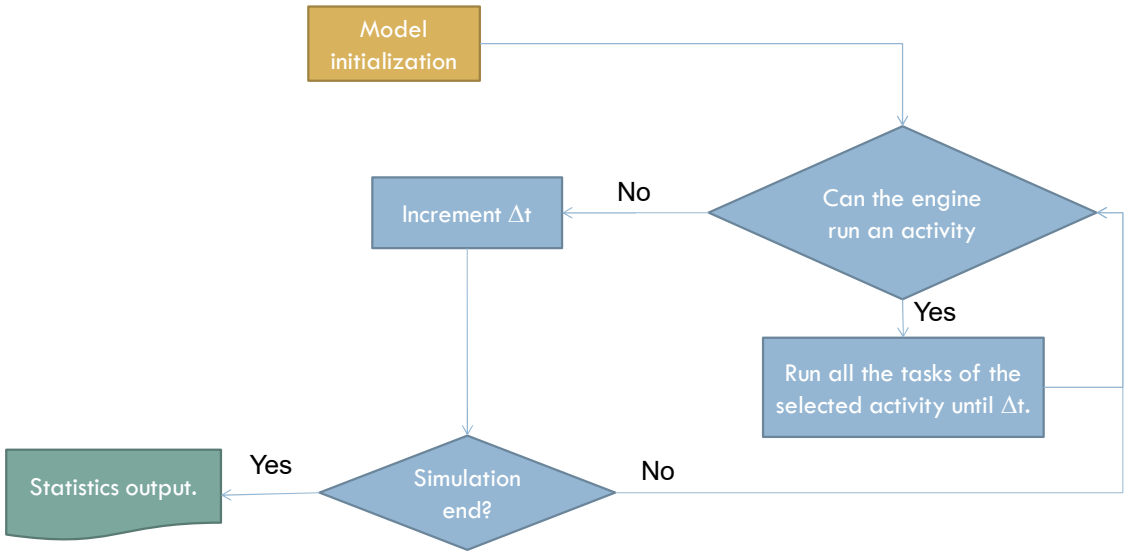
On Activity 2 Task 01 can be executed but not Task 02 that will be executed at time 7.

Activity scanning example



Finally we mark that the Activity 02 can be executed at time 7, that is the current time plus the time needed to complete the Tasks 02.

AS: simulation engine



Here you have the algorithm that represents the Activity scanning approach. We start with the model initialization, where usually the clock will be at 0. Then we detect if an Activity can be executed, if true, we will execute all the task we can. If no, we will increment the simulation clock, the Time, by delta t. Then we can detect if the finish contrition is meet. If is true we finish the simulation, if not, we will start again analyzing of other tasks can be executed.

Activity scanning Example

Trace

Now we will review a simulation trace following Activity scanning approach.

Activity scanning events evolution

- Using $\Delta t=1$. run the simulation until time = 6.

Id	Time	Event Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
1	1		1,6933	1E+12	0	0	0	0

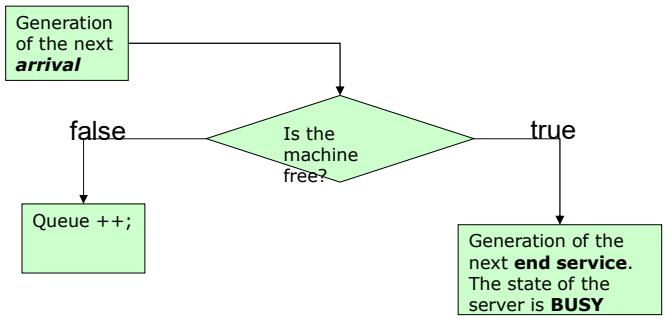
Next arrival	Next exit
1,6933	1,8840
4,0012	4,3038
5,2509	5,6282
5,5315	6,5012
5,6327	
6,0014	

This will be the start point for our simulation. We present here the table with all the events to be able to compute the same trace, in a real case the events will be calculated when needed using a specific probability distribution. notice that in this case we start with Time 1 and not 0, just as a decision to represent the initial conditions of the system.

On the trace, the column Event time is only used for our understanding on the model evolution, but do not belongs to the Activity Scanning approach.

Activity scanning events evolution

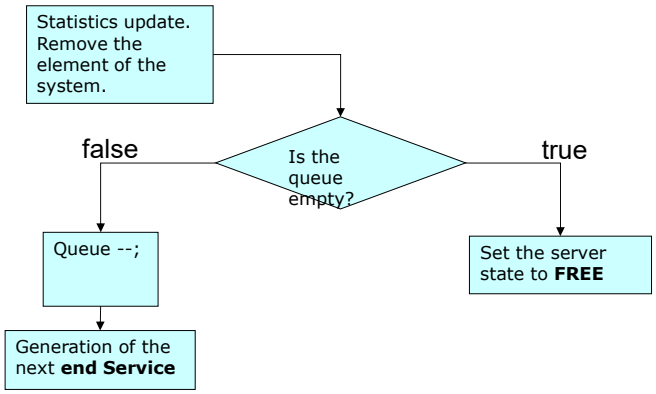
Id	Time	Event Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
1	1		1,6933	1E+12	0	0	0	0
2	2	1,6933	4,0012	1,8840	1	0	1	0



We start with the System empty, only with one event “next Arrival”. Since the time is 1 at the beginning, this event cannot be executed, hence we must increate the time using the delta t. At the second row we see that the time is updated to 2 and now, the Execution time for the Arrival event, is smaller than this time, hence, can be executed. We consider that the Activities are composed here by a single task that is the one represented in the Event Scheduling algorithm. Hence in this case we will generate a new arrival, and, because the server is free, we generate the time needed to do the service on the server. Notice that the time to do this operation is 1.8.

Activity scanning events evolution

Id	Time	Event Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
1	1		1,6933	1E+12	0	0	0	0
2	2	1,6933	4,0012	1,8840	1	0	1	0
3	2	1,8840	4,0012	1E+12	0	0	0	1



Since the time to do this operation if 1.8, at this time 2, the entity leaves the system, and is executed the Activity “Exit” that is composed by a single Task, represented on this diagram. Since the queue is empty, the state of the server now is free, represented by 0.

Activity scanning events evolution

Id	Time	Event Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
1	1		1,6933	1E+12	0	0	0	0
2	2	1,6933	4,0012	1,8840	1	0	1	0
3	2	1,8840	4,0012	1E+12	0	0	0	1
4	2		4,0012	1E+12	0	0	0	0

Notice that at time 2 nothing more can be executed; hence we must update the time using the delta t. Also notice that, from the point of view of the engine, at time two an entity enters and leaves the system, but we do not have the detail regarding the time that this entity spends on the system, since the column Event time is only used for our understanding on the model evolution, but do not belongs to the Activity Scanning approach.

Activity scanning events evolution

Id	Time	Event Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
1	1		1,6933	1E+12	0	0	0	0
2	2	1,6933	4,0012	1,8840	1	0	1	0
3	2	1,8840	4,0012	1E+12	0	0	0	1
4	2		4,0012	1E+12	0	0	0	0
5	3		4,0012	1E+12	0	0	0	0

We update Time to 3…

Activity scanning events evolution

Id	Time	Event Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
1	1		1,6933	1E+12	0	0	0	0
2	2	1,6933	4,0012	1,8840	1	0	1	0
3	2	1,8840	4,0012	1E+12	0	0	0	1
4	2		4,0012	1E+12	0	0	0	0
5	3		4,0012	1E+12	0	0	0	0
6	4		4,0012	1E+12	0	0	0	0

We update Time to 4…

Activity scanning events evolution

Id	Time	Event Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
1	1		1,69335	1E+12	0	0	0	0
2	2	1,6933	4,0012	1,8840	1	0	1	0
3	2	1,8840	4,0012	1E+12	0	0	0	1
4	2		4,0012	1E+12	0	0	0	0
5	3		4,0012	1E+12	0	0	0	0
6	4		4,0012	1E+12	0	0	0	0
7	5	4,0012	5,2509	4,3038	1	0	1	0

And finally, we update Time to 5, where we can execute again an Activity, in this case the activity related to the Arrival process. Notice that we generate a new arrival, and again, since the server is free, we can generate the exit time.

Activity scanning events evolution

Id	Time	Event Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
1	1		1,6933	1E+12	0	0	0	0
2	2	1,6933	4,0012	1,8840	1	0	1	0
3	2	1,8840	4,0012	1E+12	0	0	0	1
4	2		4,0012	1E+12	0	0	0	0
5	3		4,0012	1E+12	0	0	0	0
6	4		4,0012	1E+12	0	0	0	0
7	5	4,0012	5,2509	4,3038	1	0	1	0
8	5	4,3038	5,2509	1E+12	0	0	0	1

Like in the previous situation, now we can process the departure, and this happens also in time 5, implying again that we will lose the detail regarding the time that this entity spends on the System.

Activity scanning events evolution

Id	Time	Event Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
1	1		1,6933	1E+12	0	0	0	0
2	2	1,6933	4,0012	1,8840	1	0	1	0
3	2	1,8840	4,0012	1E+12	0	0	0	1
4	2		4,0012	1E+12	0	0	0	0
5	3		4,0012	1E+12	0	0	0	0
6	4		4,0012	1E+12	0	0	0	0
7	5	4,0012	5,2509	4,3038	1	0	1	0
8	5	4,3038	5,2509	1E+12	0	0	0	1
9	6	5,2509	5,5315	5,6282	1	0	1	0
10	6	5,5315	5,6327	5,6282	1	1	1	0
11	6	5,6282	5,6327	6,5012	1	0	0	1
12	6	5,6327	6,0014	6,5012	1	1	1	0

Finally we can continue with the execution. Notice that rows from 4 to 6 are used only to increment the Time, but from the point of view of the simulation are useless. Notice however that this is not free, since every time that we check the feasibility to execute an Activity, a set of conditions must be evaluated, and some of them maybe involves elements of the real system that needs some time to provide the answer. Hence, we desire to reduce at maximum this empty spaces, that implies increasing delta t.

However, notice that in time 2 and in time 5, there is an arrival and a departure that takes place at the same time. We will not able to detect the statistical detail of the time that this entity remains in the system with this delta T of one. In order to obtain more detail we must decrease the delta t.

As you will notice, the selection of the delta T is a key aspect that will provide more performance or more detail to or simulation results, hence is a trade off that must be accurately justified.

In the current commercial tools that uses this approach to connect with the industry the approach will be hybrid, using internally an Event Scheduling simulation engine, and over this, an Activity Scanning engine to connect the simulation with the real system in real time.

Output comparison

Compare the previous trace wit one obtained using Event Scheduling

Do this by yourself, compare the previous trace wit one obtained using Event Scheduling.

Event scheduling events evolution

Using this data

Id	Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
0	0	0	0	0	0	0	0

Next arrival	**Next exit**
1,6933	1,8840
4,0012	4,3038
5,2509	5,6282
5,5315	6,5012
5,6327	7,0477
6,0014	

Event scheduling events evolution

Id	Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
		1,6933	1E+12				

Event scheduling events evolution

Id	Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
		1,6933	1E+12				
1	1,6933	4,0012	1,8840	1	0	1	0

Event scheduling events evolution

Id	Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
		1,6933	1E+12				
1	1,6933	4,0012	1,8840	1	0	1	0
2	1,8840	4,0012	1E+12	0	0	0	1

Event scheduling events evolution

Id	Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
		1,693356	1E+12				
1	1,693356	4,001288	1,884081404	1	0	1	0
2	1,884081	4,001288	1E+12	0	0	0	1
3	4,001288	5,250927	4,303805741	1	0	1	0

Event scheduling events evolution

Id	Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
		1,6933	1E+12				
1	1,6933	4,0012	1,8840	1	0	1	0
2	1,8840	4,0012	1E+12	0	0	0	1
3	4,0012	5,2509	4,3038	1	0	1	0
4	4,3038	5,2509	1E+12	0	0	0	1

Event scheduling events evolution

Id	Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
		1,6933	1E+12				
1	1,6933	4,0012	1,8840	1	0	1	0
2	1,8840	4,0012	1E+12	0	0	0	1
3	4,0012	5,2509	4,3038	1	0	1	0
4	4,3038	5,2509	1E+12	0	0	0	1
5	5,2509	5,5315	5,6282	1	0	1	0

Event scheduling events evolution

Id	Time	Next arrival	Next exit	Server state	Queue	Arrive	Exit
		1,6933	1E+12				
1	1,6933	4,0012	1,8840	1	0	1	0
2	1,8840	4,0012	1E+12	0	0	0	1
3	4,0012	5,2509	4,3038	1	0	1	0
4	4,3038	5,2509	1E+12	0	0	0	1
5	5,2509	5,5315	5,6282	1	0	1	0
6	5,5315	5,6327	5,6282	1	1	1	0
7	5,6282	5,6327	6,5012	1	0	0	1
8	5,6327	6,0014	6,5012	1	1	1	0
9	6,0014	7,3736	6,5012	1	2	1	0
10	6,5012	7,3736	7,0477	1	1	0	1
11	7,0477			1	0	0	1

Es ok en el punt marcat, fixeu-vos que aquesta propera sortida be marcada per