



# Simulation modeling in Renque

## A simple call center model

### Introduction

This document describes the Renque model [CallCenter.zip](#), which represents a simple call center. It is intended to serve as starting point for more advanced modeling tasks involving call centers or similar systems. The text assumes that the reader has basic knowledge of simulation modeling in Renque, as covered by the *Getting started* section of the application's user manual.

### System description

A call center operates from 8 AM to 7 PM on weekdays. The incoming calls are answered by a group of operators of which the number varies by the hour. If all operators are busy a computer places incoming calls on hold in a queue with unlimited capacity.

The rate of incoming calls has been determined from operational data of the call center. The call arrival interval time is approximated by an exponential distribution with a mean that appears to vary slightly over the day. Callers will hang up the phone when the waiting time exceeds their tolerance level. The waiting time limit was found to be distributed exponentially with a mean of 3 minutes. Furthermore, three call types are distinguished with distinct processing time characteristics. Each call type has a triangularly distributed processing time, with constant minimum, modulus and maximum values.

### Model input

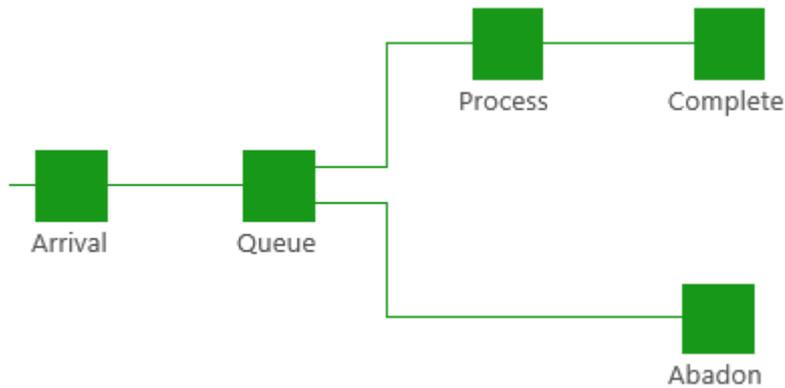
The observed data are summarized in the following input parameter spreadsheet.

	C	D	E	F	G	H	I	J	K	L	M
7											Processing time [min]
8	Clock period	Arrivals [1/hr]		Staffing [-]		Call type	frequency [%]		Mod	Min	Max
9	8:00	30		4		1	40		3.5	0	5
10	9:00	40		5		2	40		10.5	2	20
11	10:00	50		6		3	20		6.5	1	10
12	11:00	60		7							
13	12:00	60		7							
14	13:00	50		7			Hold time limit mean		3		
15	14:00	60		7							
16	15:00	60		7							
17	16:00	60		7							
18	17:00	40		7							
19	18:00			3							

Column C defines the start time of one hour clock periods during which average call rate (Column D) and the number of active operators (Column F) are assumed to be constant. The call center stops accepting incoming calls at 6 pm, but some operators will stay another hour to process the calls put on hold. Column I specifies the frequency distribution for the three call types. The processing time properties are given in Columns K-M of rows 9-11. Cell K14 specifies the mean value of the waiting time limit.

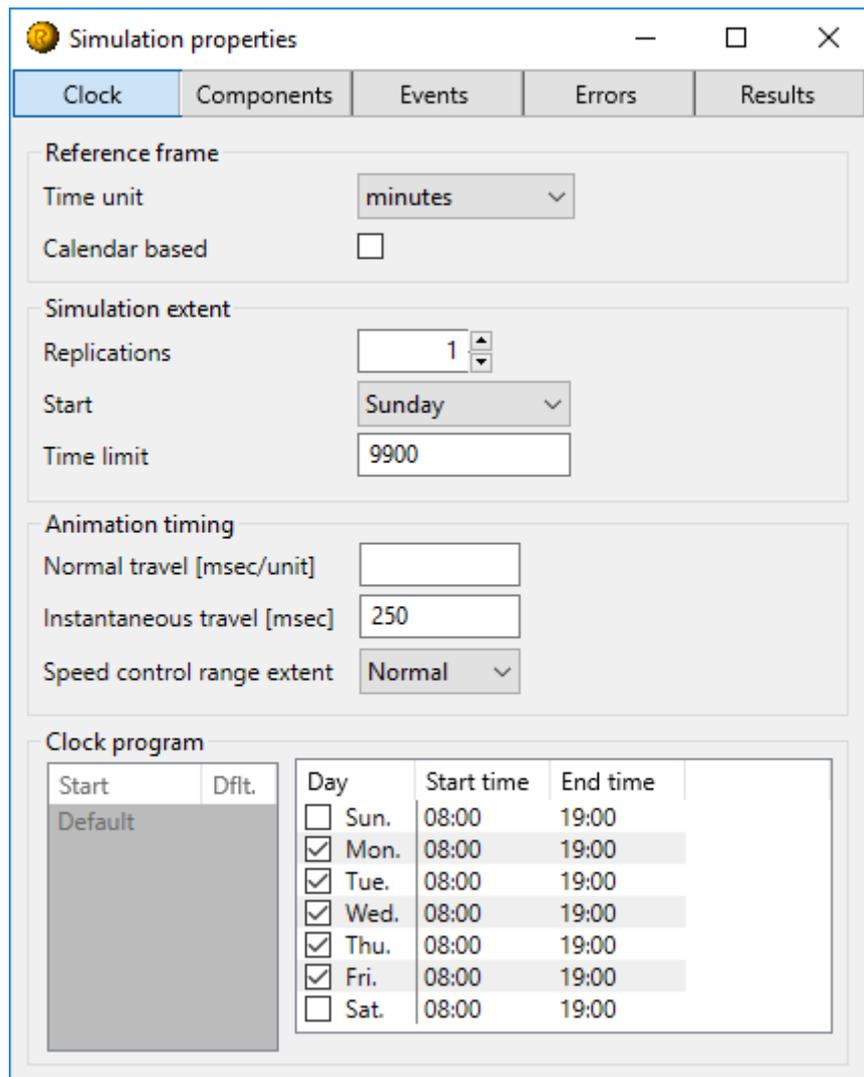
## Model description

The model consists of four interlinked servers, as displayed in the figure below. It also has two charts a number of component objects.



The server named Arrival generates a flow of entities representing the incoming calls. The calls are sent on one of the three links to the Queue server. The Process server collects these entities from the Queue server for processing. The server named Abandon receives all entities representing callers who hung up the phone.

The clock properties of the simulation model are displayed in the figure below. The model has a preset runtime of 9900 minutes (3 weeks). The simulation runs on weekdays only because the weekend days have been unchecked in the default clock program timetable.



The time-dependent input parameter spreadsheet data are incorporated into the model using two variable components. The desired call arrival rates shown in range D9-D18 of the input data spreadsheet are stored in variable NArr(0 to 9). The keys 0 - 9 each represent an hourly time period. Index 0 corresponds to the time period 8 to 9 AM. The same was done for the staffing data with the variable Nopr(0 to 10). The values of these variables are assigned in the schedule component S\_Init, which runs once at the start of the simulation.

The schedule component named S\_Main uses the array variables to modify the properties of model objects and components during the simulation. The schedule customization looks like this:

```
Dim Idx as Integer = Sim.Clock.TimeHour - 8
If Idx <> 10 Then //else Arrival stream blocked
  if Idx = 11 then Idx = 0 //correction for roundoff errors in clock
  D_arrival.Parameter(1) = 60 / NArr(Idx)
  if Idx <> 0 and NArr(Idx) <> NArr(Idx - 1) and Arrival.ResidentCount <> 0 then
    Arrival.Residents(1).Transfer Lnk2
    Arrival.Links(-1).EntityCreate
  end
End If
if Process.Capacity < Nopr(Idx) then Process.SignalSend
Process.Capacity = Nopr(Idx)
```

The parameter `Idx` assigned in the first line represents the applicable key value for the variables `Narr` and `Nopr`.

The `Delay` property of the `Arrival` server has been assigned the exponential distribution component `D_Arrival`. The schedule adjusts the mean value of `D_Arrival` every hour to maintain the specified call arrival rate characteristic by the script:

```
D_Arr.Argument(1) = 60 / Narr(Idx)
```

This script changes `Argument(1)`, which is the mean property, of the exponential distribution `D_arr` by assigning the reciprocal value of Variable `Narr(Idx)`, which denotes the number of arrivals per hour, multiplied by 60 minutes per hour. The arrival rate change thus accomplished requires an additional adjustment:

```
Arrival.Residents(1).Transfer  
Arrival.Links(-1).EntityCreate
```

This script empties the server and creates a new entity to maintain the stream for the new frequency. This step is crucial if the arrival rate radically increases. The entity removed is deleted from the model because the `Transfer` statement has no link argument.

The schedule component named `S_Check` verifies that all calls have been processed at the end of each day. If not, the simulation is paused and a runtime error is generated to inform the user of an inconsistency in the input parameters. This schedule component has a `Priority` value 1 to make sure that the customization is executed before anything else happens in the model at the same simulation time.

The `Queue` server has the following customization script:

```
If sim.Clock.TimeHour = 18 Then  
    .ArrivalRoute 0, -1  
Else  
    Dim Idx as integer  
    Dim dRnd as Double = sim.Random  
    if dRnd > 0.8 then  
        Idx = 2  
    elseif dRnd > 0.4 then  
        Idx = 1  
    end  
    end  
    CallTime = D_Proc(Idx).Evaluate  
  
    if Process.ResidentCount < Process.Capacity then  
        Route 0, 1  
    else  
        .ArrivalRoute .Delay, 2  
    end  
End if
```

The script line

```
Route 0, -1
```

immediately discards any entity representing an incoming call after 18:00 hrs. Entities arriving earlier are sent immediately towards the `Process` server if it has capacity to handle another call by the code line

```
Route 0, 1
```

Arriving entities are stored in the server for routing to the Abandon server by the `.ArrivalAccept` statement if the Process server has no capacity to accept new callers. This path represents callers abandoning the queue by hanging up the phone if not collected by the Process server. The maximum waiting time corresponds to the Delay property of the Queue server, which has been assigned the triangular distribution component `D_abandon`, which has the probability density range specified on the input spreadsheet cell range K11-M11.

Call type assignment is performed with the aid of a random value assigned to the `Idx` variable, distributed between 0, 1 and 2 in accordance with the values specified Column I of the input parameter spreadsheet. The processing time for the different call types is controlled using the attribute component **CallTime**. The Evaluate function is employed to have the attribute store a number generated by the component, instead of a reference to the distribution component object itself. The `CallTime` attribute component has been assigned to the Delay property of the Process server, which results into application of the attribute value to the server timing.

The **Process server** represents the call handling by the operators. It has the following customization:

```
if sim.SignalIsArrival then
    .ArrivalAccept
elseif Sim.SignalEntity = nil then //scripted signal
    for i as Integer = 1 to .Capacity - .ResidentCount
        if Queue.ResidentCount <> 0 then EntityProcure
        next
    else //entity departure
        if .Capacity >= .ResidentCount then
            if Queue.ResidentCount <> 0 then .EntityProcure
        end
    end
end
```

This script causes arriving entities to be routed normally. The Signal departures-option of the server casts a signal when an entity departs from the server. For the departure signal the script retrieves a single entity from the Queue server. The capacity change signal received from the `S_Main` schedule causes retrieval of as many entities from the Queue server as needed to match the capacity with the resident count. Note that a capacity decrease does not cause any departures, which is consistent with operators not instantly hanging up the phone when their shift ends.