

Vet's Clinic

A simulation of a Vet's Clinic using Discrete Event modeling

Simulation

Academic Year
2021-2022

Sara Regali

`sara.regali@studenti.unimi.it`



Contents

1	Introduction	2
2	Model	3
2.1	Specific Agents	3
2.2	Main Agent	3
2.3	Assumptions and simplification	4
2.4	Modeling the agents' behaviour	6
2.4.1	Animal	6
2.4.2	Vet and AssistantVet	6
2.5	Random Variables	7
2.5.1	Bernoulli random variables	7
2.5.2	Normal (truncated) random variables	8
2.5.3	Custom random variables	8
3	Implementation	9
3.1	Process Modeling Library	9
3.2	Experiments Framework	10
4	Experimental Analysis And Conclusions	11

1 Introduction

Effectively managing a vet's clinic can be challenging for different reasons. First of all, there is a maximum number of vets that can work at the clinic because of the limited spaces and rooms in the building itself (visiting rooms, exam rooms, surgery room etc.), and not all vets are specialized and can deal with all of the cases, meaning that some surgeries can be performed only by specialized doctors. Furthermore, the procedure used to tackle emergency patients is very different and takes way more time than dealing with routine patients; moreover, surgery, visiting and hospitalization time can vary a lot from patient to patient, keeping doctors occupied and unable to visit other patients. There is also a maximum number of patients that can be queued at anytime, so new patients that arrive at the clinic when the queue is full will be denied access to the waiting room and sent to some other clinic. The variability of time is a serious problem linked with the uncertainty of the gravity of the patient, in fact other patients in queue may have their visit seriously delayed because of the time needed to visit and operate on accepted patients and even die because of the long wait. Another problem is that of emergency patients that have been accepted to the waiting room but need immediate medical attention (are therefore emergencies). These patients need to wait for the next doctor to become available, but that might be too late based on the gravity of their condition.

To deal with these uncertainties and to try to give the issues described a more structured approach, the problem depicted was modeled in a simulation using the Anylogic tool through the Discrete Event modeling paradigm, with the aim of collecting meaningful values for the parameters of interest, which are the amount of patients lost on average at a working day at the clinic, and the average time passed between the end of the shift and the dismissal of all patients that were inside and in line at the clinic when the shift ended.

The project's code can be found at **this link**. The following is a description of the model developed.

2 Model

The simulation paradigm chosen to represent the problem previously described was the **Discrete Events Modeling**. With this type of modeling, the focus is set on specific events that happen at a certain time, and whenever an event occurs, some relevant data are collected and statistics are made over the collected data. This type of modeling fits particularly well with the problem described as it can be easily seen as a series of interesting events, interacting with one another in a queuing system.

2.1 Specific Agents

The agents foreseen for the model are three:

- **Animal**: this is the agent used to model the patient's behaviour. It has some private parameters like **isEmergency**, **needsExams**, and **needsSurgery** that are used to determine whether a patient is considered an emergency, and if it is, whether it needs additional exams and if it needs surgery. All accepted animals are visited but only emergency patients may need exams and surgery and are hospitalized, while regular patients stay in the clinic only for the time of their visit and then immediately leave the system. The *Animal* agent also has two additional variables (**vet** and **assistant**) to keep track of the doctor to whom it was assigned to for the visit.
- **Vet**: this is the agent used to model the behaviour of specialized doctors. In this model, these doctors are preferred (but not bounded) to visit emergency patients and can perform exams and surgery on the visited patient. In order to start the surgery, an assistant is needed.
- **AssistantVet**: this is the agent used to model the behaviour of doctors that are not specialized. In this model, these doctors are preferred (but not bounded) to visit non-emergency patients and to tackle non important tasks such as putting patients in hospitalization room and dismiss them. They can also perform exams on emergency patients if needed but they need to cooperate with a vet in order to execute surgery.

2.2 Main Agent

Other than the agents specific for the problem, any Anylogic project must provide an additional agent where the simulation will actually take place. This is the Main agent, which can be fully customized to explicit how agents interact with one another and to set the problem's features of interest. The Main agent for the project was designed in the following way:

- **Parameters**: project's attributes defining some key characteristics of the problem modeled. These will be changed by the user through the experiment's presentation page.
 - **maxWaitingPatients**: parameter used to set the maximum number of waiting patients at the same time at the clinic. In this model it can vary from two to five and it's used to set the capacity of the *animalQueue* object.
 - **numberOfVets**: parameter used to set the number of Vet agents in the clinic. This can vary from one to a maximum of four and is used to set the capacity of the *resourceVet* object.

- **numberOfAssistantVets**: parameter used to set the number of AssistantVet agents in the clinic. This can vary from one to a maximum of six and is used to set the capacity of the *resourceAssistantVet* object.
- **Variables**: project's attributes used to store the results of the simulation.
 - **newPatientsDeclined**: variable used to store the number of new patients that tried to enter the clinic but were not accepted due to the waiting patient's queue being full.
 - **emergencyPatientsLost**: variable used to keep track of the emergency patients accepted to the clinic but that waited too much to be visited and therefore exited the system.
 - **timeAfterEndShift**: variable used to keep track of the time passed after the end of the shift and the moment the last patient admitted to the clinic was dismissed.
- **Constants**: useful constant variables that can be modified in case of need.
 - **NEWPATIENTSARRIVALRATE**: constant variable defining the arrival rate of new patients. In the model is set to 0.1, which means there is a new patient arriving at the clinic every 10 minutes.
 - **MAXTIMEOUTEMERGENCIES**: constant variable used to set the number of minutes an emergency patient can wait at most while in queue. In this model it's set to 30 minutes, after which the emergency patient will leave the system.
 - **ENDSHIFTMINUTES**: constant variable used to set the number of minutes after which the shift ends. In this model it's set to 720 (12 hours).
- **Events**: objects used to schedule some actions in the model.
 - **endOfShiftEvent**: event used to signal the end of the shift. It is triggered after **ENDSHIFTMINUTES** and it's used to simulate the closing of the clinic. It sets the rate of new animals arriving to zero and restarts the **stopSimulationEvent**.
 - **stopSimulationEvent**: event used to stop the simulation by calling `getEngine().stop()`. It is triggered once all admitted patients are dismissed after the closing of the clinic.

2.3 Assumptions and simplification

The simulation was based on datas collected after an interview with an actual veterinarian, but some assumptions and simplifications were made in order to lessen the model's complexity:

- Patients are already designated as emergency or not before their arrival at the clinic. This is a simplification of reality because as a matter of fact, triage is needed to identify the gravity of the patient. Moreover, the whole concept of "emergency patient" is a further simplification because there are various degrees to the gravity of a patient, and simply classifying one as an "emergency" is not enough, because not all emergencies have the same priority. Nonetheless for the sake of these experiments, patients will be cataloged as regular or emergencies;
- In the clinic modeled there can be at most ten doctors that can work in it, with at most four specialized doctors (Vet agents), and at most six doctors without a specialization (AssistantVet agents);
- In the waiting room there can be a minimum of two to at most five patients waiting;

- New emergency patients have priority over new regular patients and will be visited before them;
- The emergency patients accepted at the clinic that are in line will leave after waiting for more than 30 minutes;
- Regular patients accepted at the clinic but in queue can wait an unlimited amount of time before being visited;
- Surgery must be performed by one vet and one assistant together;
- Surgery has the highest priority over all other tasks, followed by visiting emergency patients, then dismissing patients from hospitalization, and then visiting regular patients at last;
- Hospitalized patients don't need to be monitored and will be dismissed after the hospitalization time.
- For the sake of the experiment, hospitalization time will be reduced to 75%, otherwise it would be impossible to finish the simulation in one shift for sometimes patients are hospitalized for up to four days.

2.4 Modeling the agents' behaviour

2.4.1 Animal

The following flowchart shows the behaviour of the Animal agent.

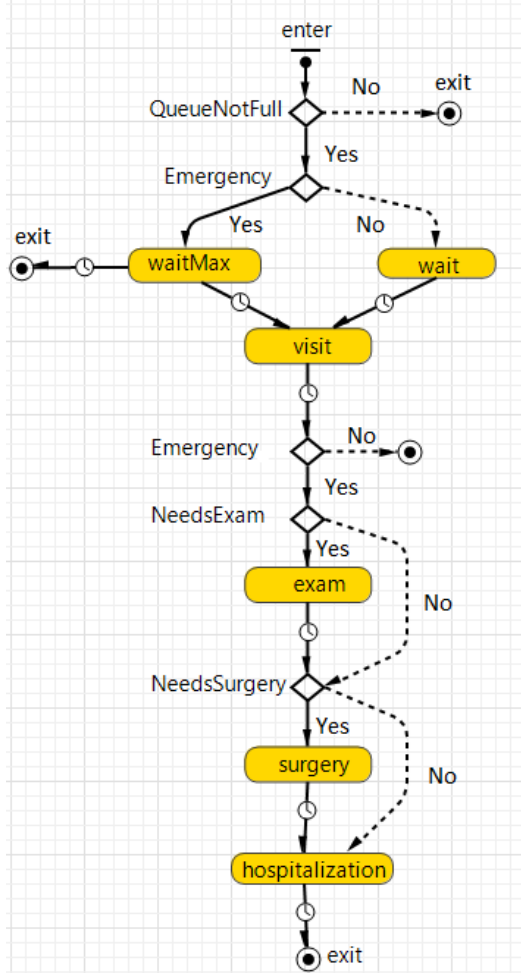


Figure 1: Diagram of the Animal agent's behaviour

When a patient enters the system, if the queue of waiting patients is full, it immediately exits; otherwise it is accepted to the waiting room. Here patients can wait for an arbitrary period of time before being visited depending on the availability of doctors. If the patient is an emergency though, it can only wait for a maximum period of time. This is done to simulate the urgency and the fact that if the patient waits for too much and is not treated, it may die.

Eventually patients in queue are visited and if they are regular patients, they are then dismissed and leave the system. After emergency patients are visited, they may need further examination (blood tests, ultrasound, x-rays etc.) and subsequently surgery. Exams are not mandatory and neither is surgery because sometimes the first visit is enough to stabilize the patient.

The last step required for emergency patients is hospitalization, after which they are dismissed and leave the system.

2.4.2 Vet and AssistantVet

The following flowcharts shows the behaviour of the Vet and AssistantVet agents.

The two agent's behaviours are very similar and differ only for two transitions. These agents enter the system at its start can't leave it. They will stay in the idle state until a new patient arrives or an hospitalized one needs to be dismissed. After a new patient is visited, the agents can either go back to the idle state if it wasn't an emergency, or perform further examinations and/or surgery in case of need.

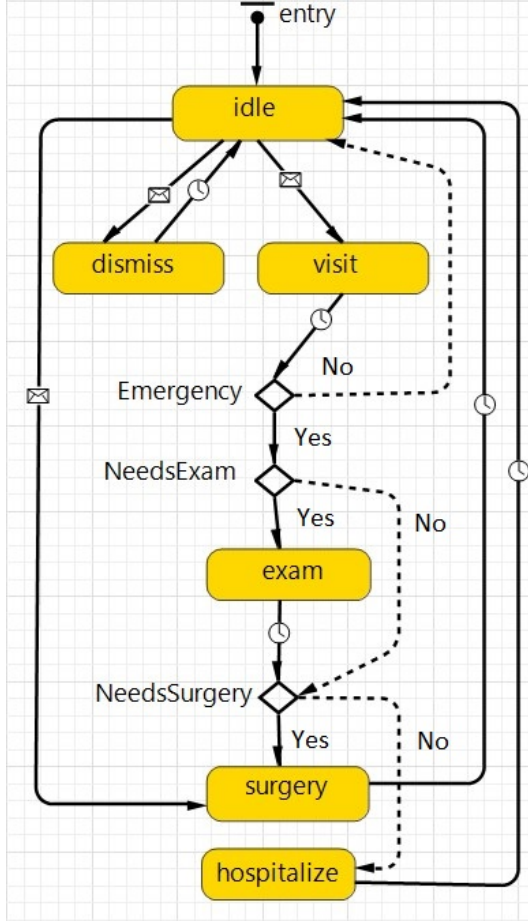


Figure 2: Diagram of the Vet agent's behaviour

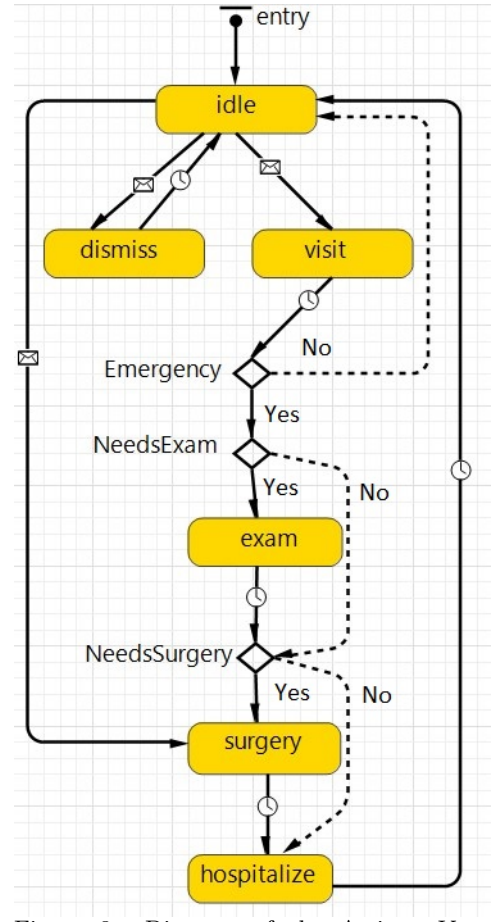


Figure 3: Diagram of the AssistantVet agent's behaviour

If surgery is needed, the Vet agent will go back to the idle state once it's finished, while the AssistantVet agent will hospitalize the operated patient. That's because the hospitalize task is a minor task and an assistant is preferred when it comes to tackling it. Vet agents can also hospitalize patients but only those that don't need surgery, because since they were already assigned to those patients, it's more efficient if they hospitalize them instead of waiting for an assistant to be available to do it. Both agents will go back to the idle state after the hospitalization task is finished.

2.5 Random Variables

2.5.1 Bernoulli random variables

A Bernoulli random variable is a discrete random variable measured in one trial for which the probability is $P(X = 1) = p = 1 - P(X = 0) = 1 - q$. In this model, the three variables **isEmergency**, **needsExams**, and **needsSurgery** of the Animal agent were designed as Bernoulli random variables with the following probability:

$$X = \begin{cases} 1 & \text{if } p < 0.5 \\ 0 & \text{otherwise} \end{cases}$$

where p is random number taken from a uniform distribution between $[0,1)$.

2.5.2 Normal (truncated) random variables

A truncated normal random variable is derived from a normally distributed random variable by bounding it to fit in the specified interval $[\text{min}, \text{max}]$, shifting it to the right by **shift** and then stretching it by the **stretch** coefficient. Based on the information acquired by the interview with the veterinarian, the random variable representing the visiting time was defined with the following parameters, using the `normal(min,max,shift,stretch)` function from Anylogic's probability distribution functions:

`normal(5,15,10,5)`

Meaning that the visit time follows a normal distribution function with $\mu = 10$ and $\sigma^2 = 5$ and may take values from 5 to 15 minutes.

The second truncated random variable determined for the model is the one representing the time taken to perform exams on an emergency patient. This one takes values ranging from 30 to 60 minutes and has μ and σ^2 set to 45 and 15 respectively, as described in the next line of code:

`normal(30,60,45,15)`

The last model parameter defined as a truncated random variable is the hospitalization time of operated patients, which varies from 60 to 180 minutes, with μ set to 120 and σ^2 set to 30. The Anylogic function used to model it was the following:

`normal(60,180,120,30)`

2.5.3 Custom random variables

There is one last random variable to model, which is the surgery time random variable. This variable doesn't fit in any of the most known distributions because of its specificity, so a custom random variable is needed to shape its behaviour on Anylogic. The following is the probability distribution function built from the observation of actual surgery times.

$$X = \begin{cases} [0; 30) & 5\% \\ [30; 45) & 25\% \\ [45; 60) & 35\% \\ [60; 90) & 15\% \\ [90; 120) & 10\% \\ [120; 150) & 5\% \\ [150; 180] & 5\% \end{cases} \quad (1)$$

3 Implementation

Most of the relevant details of how the model was implemented have already been discussed (the four agents, the parameters, variables and constants of the model and the random variables identified), but some technicalities of how the simulation was actually designed are worth mentioning.

3.1 Process Modeling Library

The most useful component of Anylogic has been the Process Modeling Library, which provides blocks and connectors to model the agent's behaviour and their interactions. Below is a diagram of the model's whole functionality.

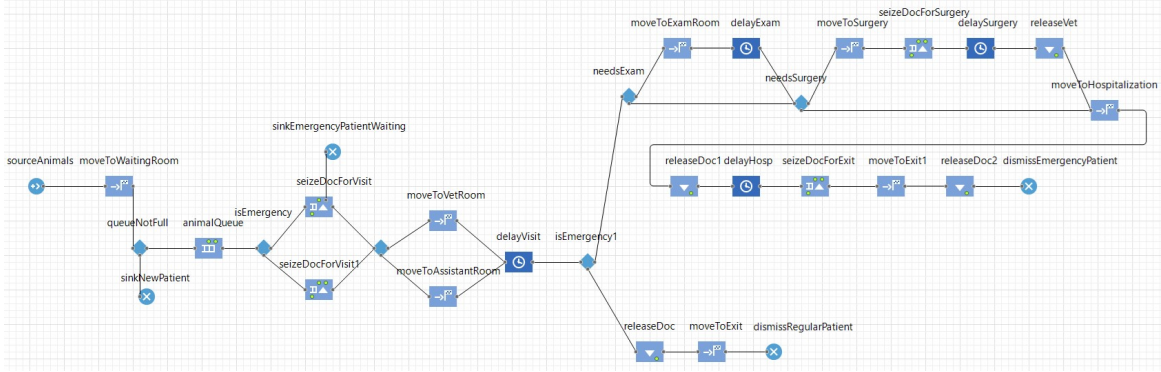








Figure 4: Diagram of the problem made with the Process modeling Library components

-  : The source component is the one responsible for simulating the arrival of new patients at a fixed rate. In the model this rate is set to `NEWPATIENTSARRIVALRATE`;
-  : The sink component simulates the departure of new patients that arrive at the clinic but can't be accepted because the queue is full, the departure of new emergency patients that wait in line for too long, and the dismissal of hospitalized patients.
-  : The queue block was used to manage the queue of waiting patients. It was defined as a FIFO queue and its capacity set to the variable `MAXWAITINGPATIENTS`;
-  : The delay component simulates the duration required to visit, evaluate, and operate on a patient with the delay time set as one of the random variables previously discussed.
-  : The seize component is particularly useful to simulate a patient's need for a doctor. Here it was also used to set the priorities among the four main tasks (visit of emergency patient, visit of regular patient, surgery and dismissal of operated patient) in the `Priorities\Task priority` section to explicit the fact that emergency patients in queue need to be visited before regular patients, and that the surgery task needs to be tackled before all other tasks, followed by visiting emergency patients, then dismissing patients from hospitalization, and then visiting regular patients at last. Furthermore, it proposes an `Enable exit on timeout` option in the `Advanced` section that was used to model the exit of emergency patients in queue that have waited for too long.
-  : The release block was used to model the finishing of the task by the seized doctor.

3.2 Experiments Framework

In order to allow for an easy modeling and testing of the problem identified, Anylogic offers various architectures for setting up experiments, one of which being the **Simulation** capability, which performs model runs for the required number of times, with the specified parameters.

Through the **Properties** tab of the experiment, some useful features of the experiment itself were set to perform our modeling, for example:

- **Model time tab:** the Execution Mode was set to **Virtual time (as fast as possible)**, making the model run at its maximum speed without mapping the model's time to real time.
- **Randomness tab:** the Random seed (unique simulation runs) option was selected to make sure each simulation run is unique.
- **Java actions tab:** in the **After simulation run** section, a small function was designed to repeat each run 365 times and simulate a whole year of experiments:

```
if(getEngine().getRunCount()<365){  
    run();  
}else{  
    stop();  
}
```

The function described above is executed after a single run ends, and stops the whole experiment when the engine's `runCount` gets to 365.

Furthermore, by unchecking the **Skip experiment screen and run the model** box, it's possible to set up a user interface to allow the user to set the values of the parameters of interest before running the model. In the project, this was achieved by the use of three sliders, one to set each parameter described in subsection 2.2; moreover a button was added to allow the user to start the experiment after setting those parameters.

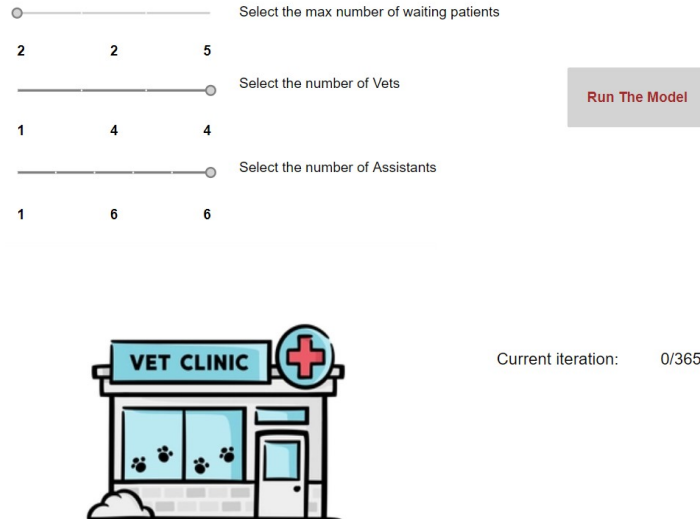


Figure 5: Screenshot of the experiment's presentation page

4 Experimental Analysis And Conclusions

The Simulation experiment described before was run different times, one for each significant combination of the parameters described in subsection 2.2. Each experiment consists of 365 shifts (i.e. days) at the clinic, as was previously mentioned. Below are the experimental results found.

#	Queue capacity	Vets	Assistant Vets	Sum Of Doctors	Average New Patients Lost	Average Emergency Patients Lost	Average Time After End Shift
1	4	3	3	6	9.39	2.23	240.63
2	4	3	4	7	5.84	1.45	233.89
3	4	3	5	8	3.01	0.75	235.51
4	4	3	6	9	1.85	0.45	233.17
5	4	4	4	8	3.28	0.90	232.32
6	4	4	5	9	1.61	0.47	230.26
7	4	4	6	10	0.78	0.26	228.52
8	5	4	4	8	2.35	0.89	230.15
9	5	4	5	9	1.15	0.47	226.10
10	5	4	6	10	0.67	0.27	227.18

Table 1: Results of the experiment's run

As one may expect, the sum of the average number of new patients lost and the average number of emergency patients lost decreases as the sum of doctors increases. Besides that, there are a few interesting facts emerging from the table above:

1. Being the number of doctors and the queue capacity equal, better results are achieved when the number of assistants is greater than the number of vets, in terms of total patients loss (rows 3 and 5).
2. Being the combination of doctors equal but having an increased queue capacity, better results are achieved when the queue capacity is higher, in terms of new patients loss (rows 5 and 8; 6 and 9; 7 and 10).
3. With enough doctors and being their number equal, the average of emergency patients lost remains constant even while increasing the queue capacity.

The first fact can be explained by the decisions made while modeling the system. Because of the fact that surgery must be performed both by a Vet and an AssistantVet agents together, in those cases where there are more assistants than vets, those agents that are not busy with surgery will be able to assist waiting patients, decreasing the total loss of patients. Of course as the number of vets increases there can be more teams performing surgery at the same time, leading to a lower number of patients lost on average (this can be seen comparing rows 4 and 6.)

The second fact is not as straightforward but can be easily explained. If the capacity of the queue is increased, a small part of those new patients that would have immediately left the system before, can now enter the queue and have a chance to be visited, eventually leading to a smaller total average of new patients lost.

Fact three seems to go against fact two, but since the probability of a patient of being emergency or not stays the same, it follows that the average number of emergency patients lost isn't affected by the increasing capacity of the queue.

In conclusion, while the average time spent at the clinic after the end of the shift can be estimated to be 230 minutes (almost four hours), the average number of patients lost in total is strictly linked to the number of doctors available, and how the different types of doctors combine. Undoubtely though, as seen in the experiment's results, increasing the capacity of the queue and admitting more patients in the waiting room when possible, will lead to a decreasing number of patients lost on average.

References

- [1] T. A. Company, Anylogic documentation (2000).
URL <https://anylogic.help/>
- [2] S. Ross, Simulation, Elsevier Inc., 2013, 5th Edition.