TRANSPORT AND
TELECOMMUNICATION
INSTITUTE

System Analysis and Modelling

Practical Assignment 1

Student's:
Artyom Kabish
(ST81578),

RIGA
2025

Table of Contents

Description of the system and conceptual model

We consider a server that processes jobs of two types. Both jobs enter the server with inter-arrival time under different probability distributions specified, according to the individual variant (I1 = Exponential (1.5) and I2 = Exponential (4)). Additionally, it takes the server a different amount of time to process the job depending on its type (P1 = Normal (2, 0.3) and P2 = Normal (2.5, 0.1)). The server can process only one job at the same time. If jobs are arriving faster than can be processed by the server, they form a queue. As soon as the server finishes the processing, the next job is taken from the queue, according to a discipline mentioned in the individual variant (Q = FIFO). The modelling time is 500 minutes. The MoE (measures of effectiveness) are indicated in the individual variant (MoE1 = Downtime factor and MoE2 = Max. of all jobs in queue).

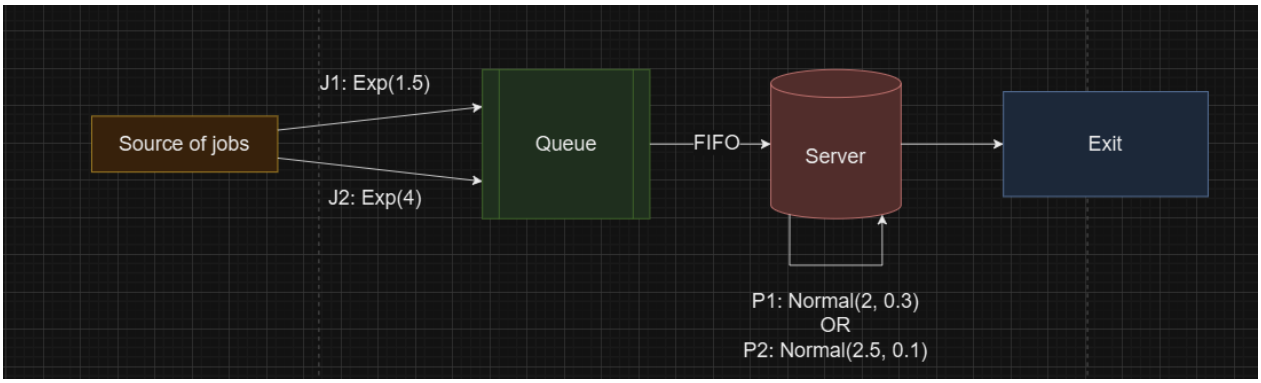

Fig. 1 – Conceptual diagram of the system

Model 1 Developing (Programming language)

Flowchart algorithm of the model

Algorithm based on sequential event manipulation. Firstly it's required to initialize simulation by generating time of appering tasks. Next, the algorithm enter the loop that will cycled while the time will be not more than 500. The loop can be leaved than total time is bigger then 500, but the additional check is added to skip all operations then the next event time is bigger then 500 and it's replace the previous total time.

At each iteration, the specific event is happenes depends on arrival time or operating time. In cases then the minimal value is one of the arrival time then happenes query operating described on the Flowchart (Fig.2). The exponential random value generator with specific parameters depends on the type of task that should be arrived used for generating arriving time while normal random generator used for generating operation time depends on task type.

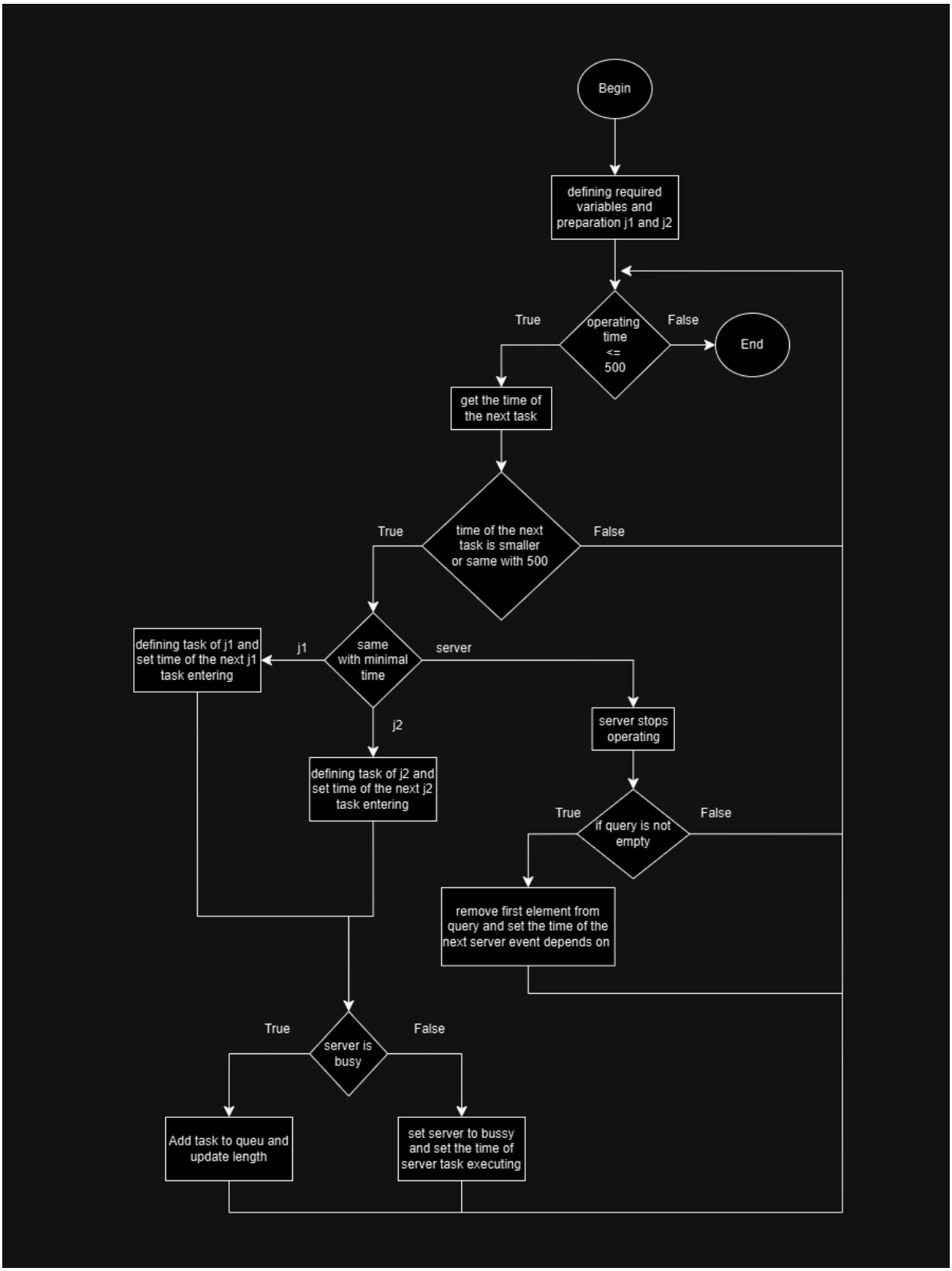

Fig. 2 – Flowchart algorithm for Model 1

Model description

The model 1 is developed using Python programmin language and follow the algorithm descirbied in the previouse section(Fig. 2).

The simulation runs for 500 time units, during one iteration the 1 out of 3 events is happened and it logs the event name, time units of task arriving, time units of server execution finish, server status, max length of query and array of query tasks FIFO(last 20 records will not show the query because the size of it is to big).

The **“exponential_random”** and **“normal_random”** functions are used for generating a random values from exponential and normal distributions. These function receive different parameters depends on the type of task and used for generating apperance time and executing time for j1 and j2 tasks.

The **“queryManager”** function operate with query when the event arriving. If the server is not bussy then arrived task automatically go to server and set the execution time that server will be bussy else the arrived task adds to the query.

In the loop firsly change server status to free and next, happenes selection of the event. It happenes by next execution time of current event. The minimal value from time units of the variables with time of next arrival j1 and j2 tasks, and server task execution finish. Received value compares with provided 3 time units and by that comparision select the next set of actions.

The **“queryManager”** function will be called in cases of arrival task events, but the main difference of different events arrival is the parameter that enter in the call method of **“queryManager”**.

But set of actions is different for server task execution finish event. The programm check the size of actual queue and if the size of queu is not 0 then the generated executing time sum with previous time of current server task execution finish time and set time as busy. In another case, the value of the task execution finish change to 500 (to handle situations then query is empty and server finished his tasks, to not give the algorithm to be sicled)

Random values generator testing

The specific python script is developed for generating a samples of random values generated by custom functions used in the fisrt model. That script placed in the another file and call the functions of custom random values generators by importing it from the main file of model 1. The generated file contains one sample generated by normal random values generator and sample generated by exponential random values generator. As parameters for random generators the values provided by personal task is used for generating 500 random values of each sample.

File has a type of csv and it used in the script of R language.The theoretical value generates by default function of R language.

Hypothesies:

- Kolmogorov-smirnov test:
    - Null hypothesis: The sample follows exponential distribution
    - Alternative: The sample distribution is different to exponential distribution
    - Significance level: 5% or 0.05
- Chi-square test:
    - Null hypothesis: The difference between two samples is acceptable to say that these samples follow same distribution
    - Alternative: The difference between two samples is obviously and all coincidences are accidental
    - Significance level: 5% or 0.05

Testing exponential random generator (rate = 1.5)

	Rate(λ)	Mean
Emperical	1.4723607	0.6791814
Theoretical	1.5	0.6666667

Table 1. emperical end theoretical values for exponential random generator with rate 1.5

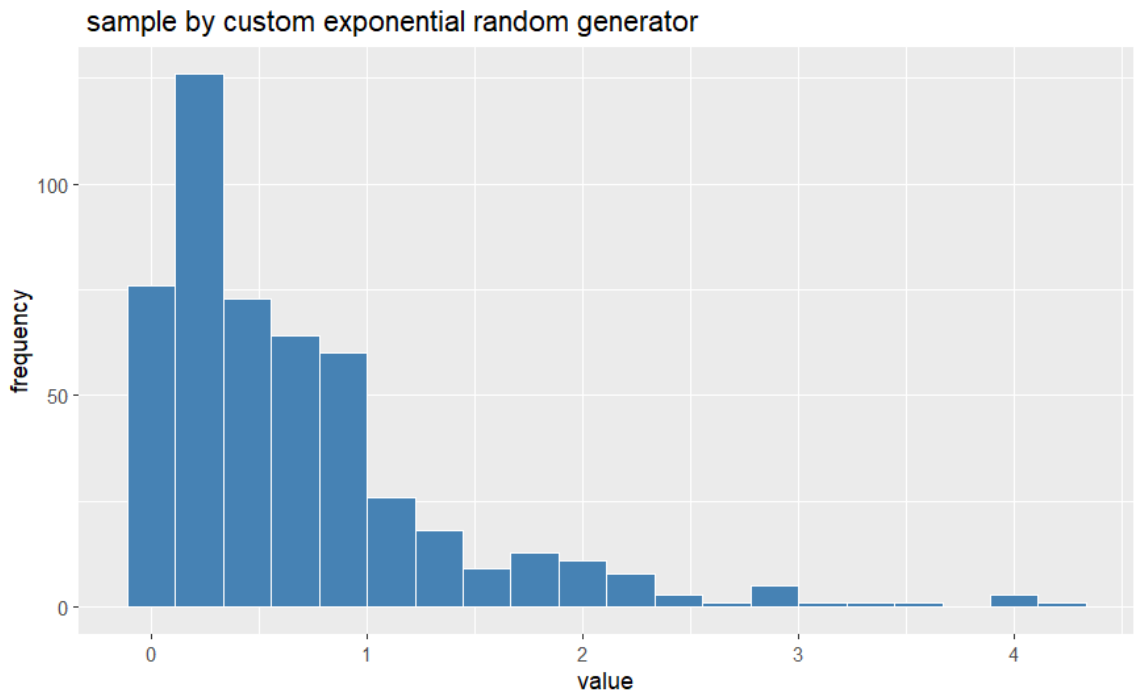

Figure 1. histogram of exponential random sample from python

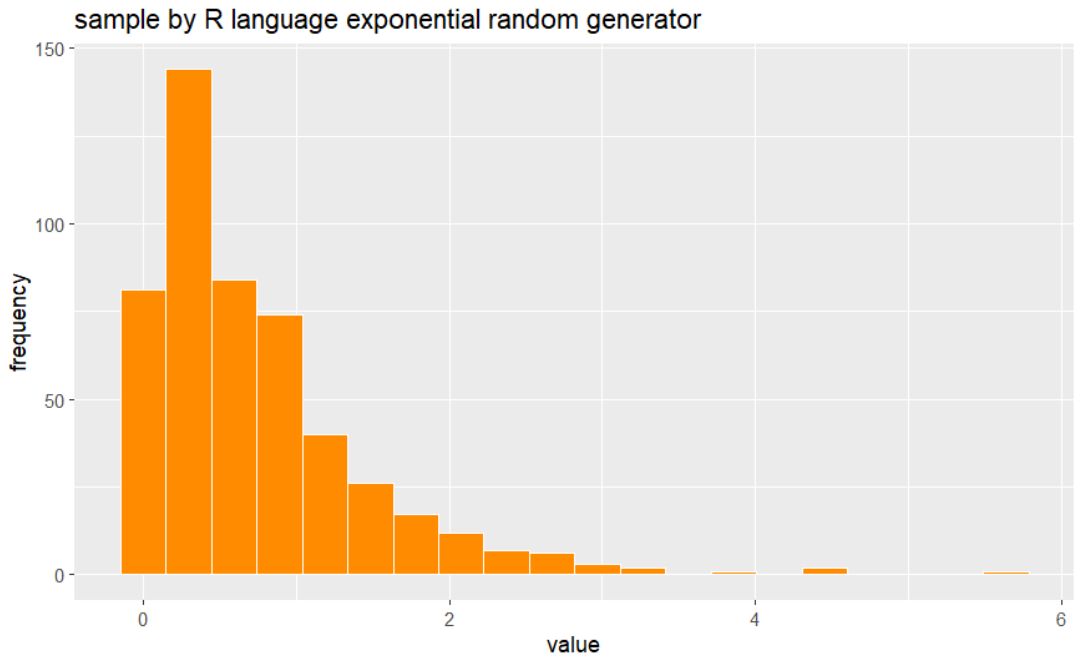

Figure 2. histogram of exponential random sample from R

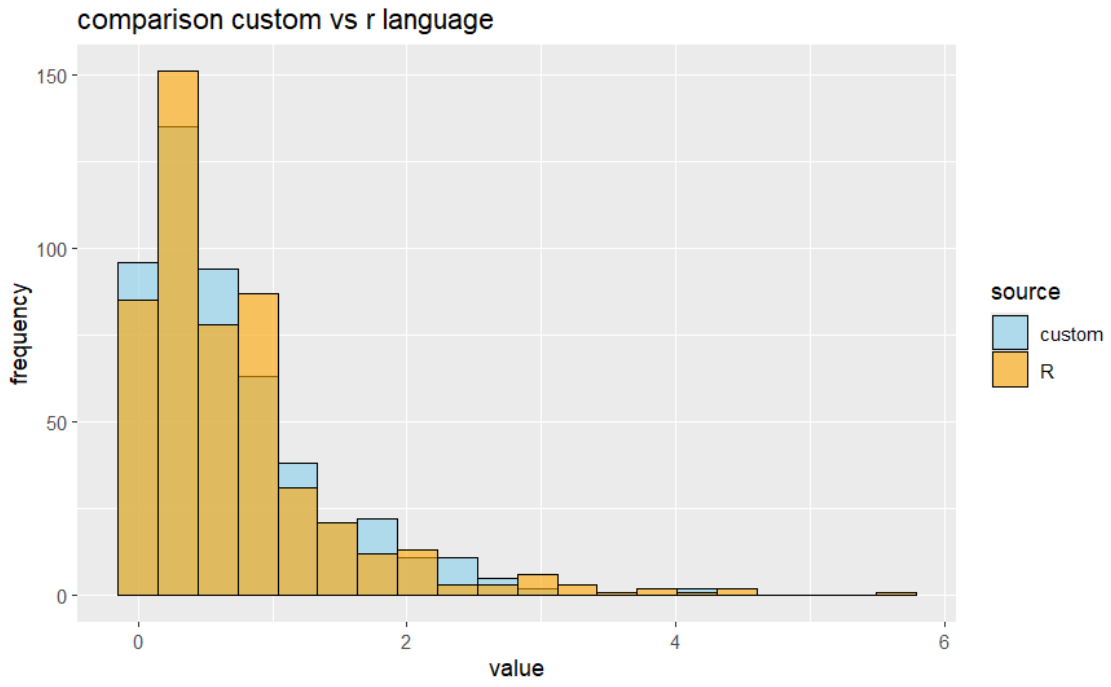

Figure 3. comparision of histograms of exponential random sample from python and R

- Kolmogorov-Smirnov test

```
        Asymptotic two-sample Kolmogorov-Smirnov test

data:  exp_emperical and exp_theoretical
D = 0.064, p-value = 0.2574
alternative hypothesis: two-sided
```

Figure 4. result of Kolmogorov-Smirnov test with exponential samples with rate 1.5

Conclusion: The p-value is bigger then 0.05 that means that Null hypothesis is successful and it means that the sample follow exponential distribution

- Chi-square test

```
        Pearson's Chi-squared test

data:  freq_emperical and freq_theoretical
X-squared = 60, df = 48, p-value = 0.1146
```

Figure 5. result of Chi-square test with exponential samples with rate 1.5

Conclusion: The p-value is bigger then 0.05 that means that Null hypothesis is successful and it means that the samples are similar and it means that they follow same distribution

Testing exponential random generator (rate = 4)

	Rate(λ)	Mean
Emperical	4.3309748	0.2308949
Theoretical	4	0.25

Table 2. emperical end theoretical values for exponential random generator with rate 4

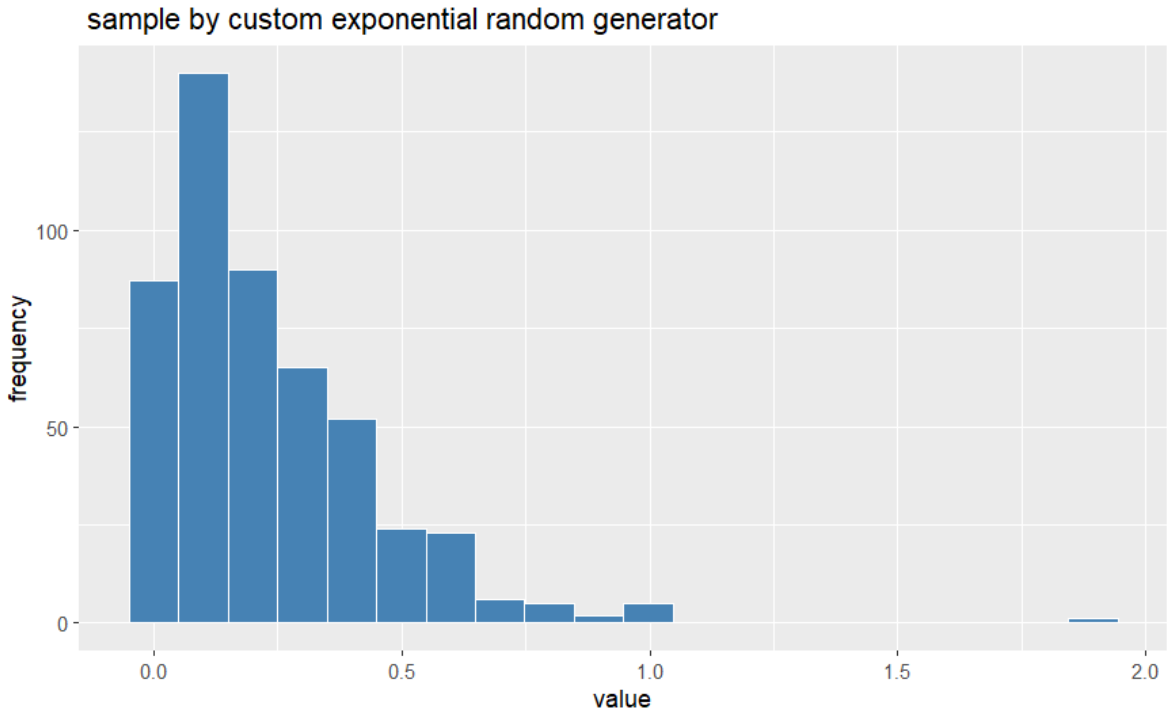

Figure 6. histogram of exponential random sample from python

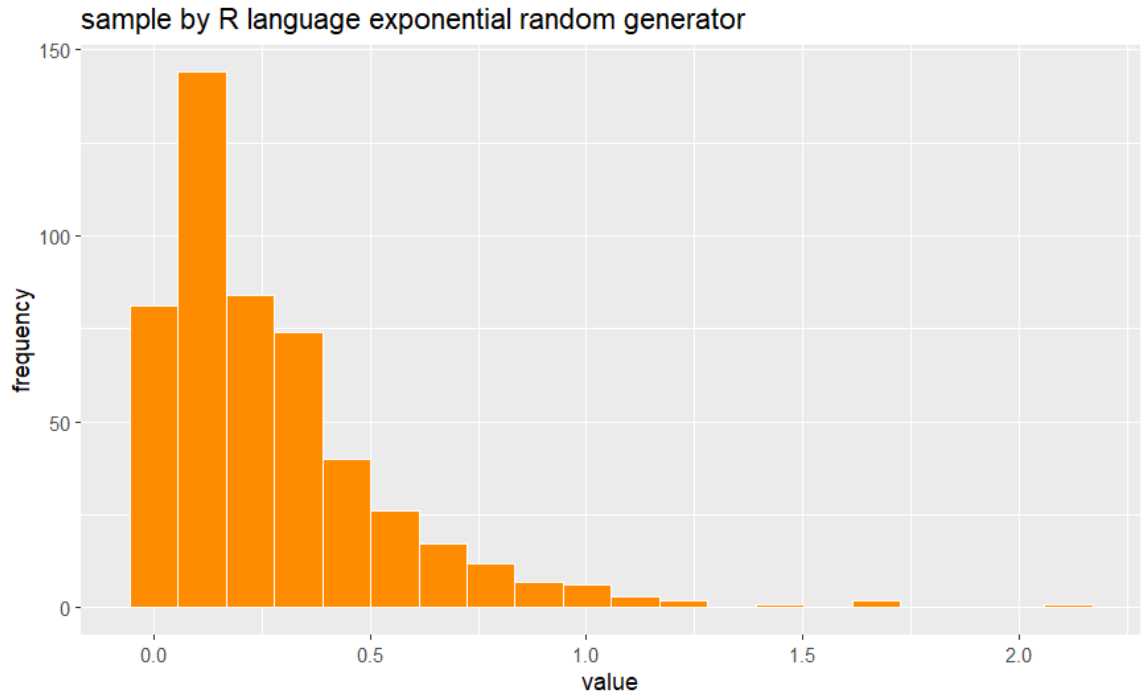

Figure 7. histogram of exponential random sample from R

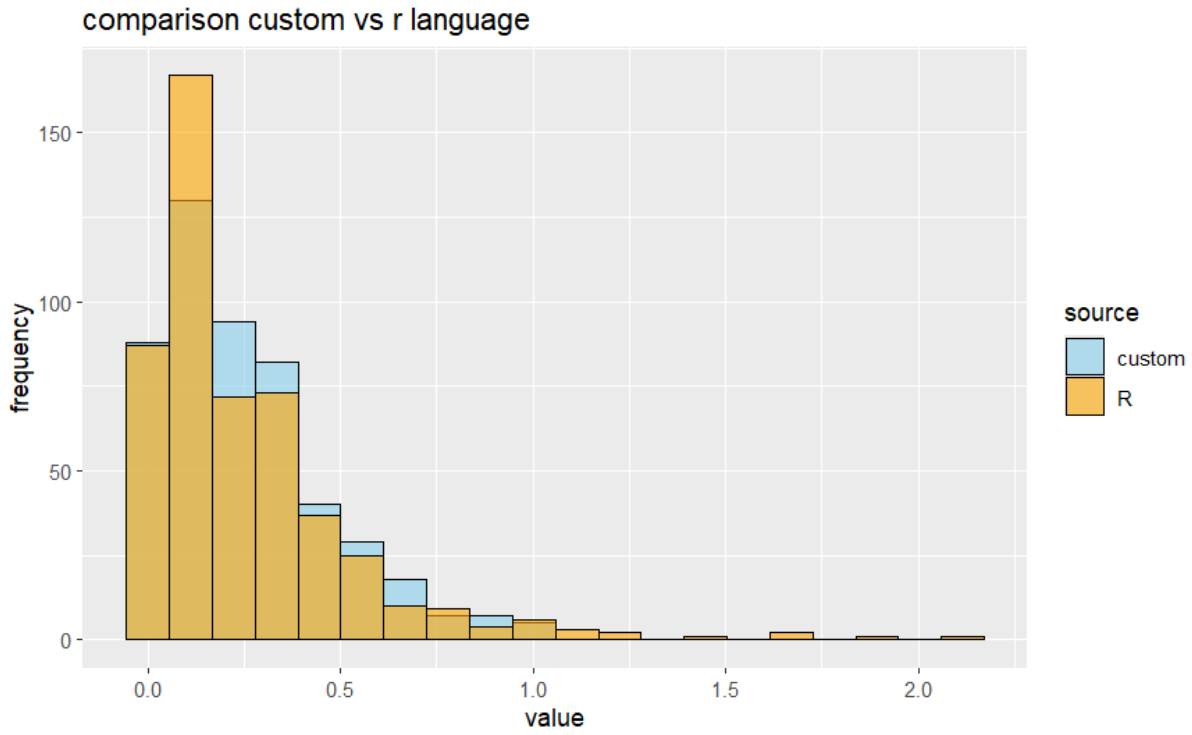

Figure 8. comparision of histograms of exponential random sample from python and R

- Kolmogorov-Smirnov test

```
        Asymptotic two-sample Kolmogorov-Smirnov test

data:  exp_emperical and exp_theoretical
D = 0.076, p-value = 0.1114
alternative hypothesis: two-sided
```

Figure 9. result of Kolmogorov-Smirnov test with exponential samples with rate 4

Conclusion: The p-value is bigger then 0.05 that means that Null hypothesis is successful and it means that the sample follow exponential distribution

- Chi-square test

```
        Pearson's Chi-squared test

data:  freq_emperical and freq_theoretical
X-squared = 50.625, df = 36, p-value = 0.05372
```

Figure 10. result of Chi-square test with exponential samples with rate 4

Conclusion: The p-value is bigger then 0.05 that means that Null hypothesis is successful and it means that the samples are similar and it means that they follow same distribution

Testing normal random generator (mean=2;sd=0.3)

	Standard Deviation	Mean
Emperical	0.2997583	1.984217
Theoretical	0.3	2

Table 3. emperical end theoretical values for exponential random generator with rate 4

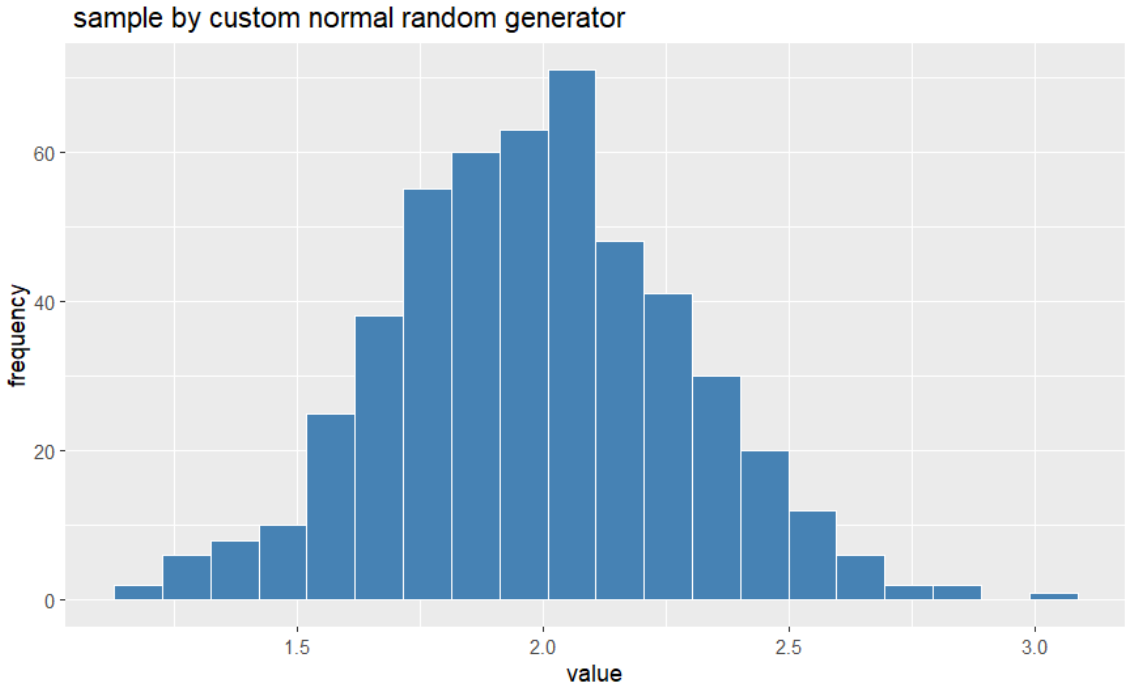

Figure 11. histogram of normal random sample from python

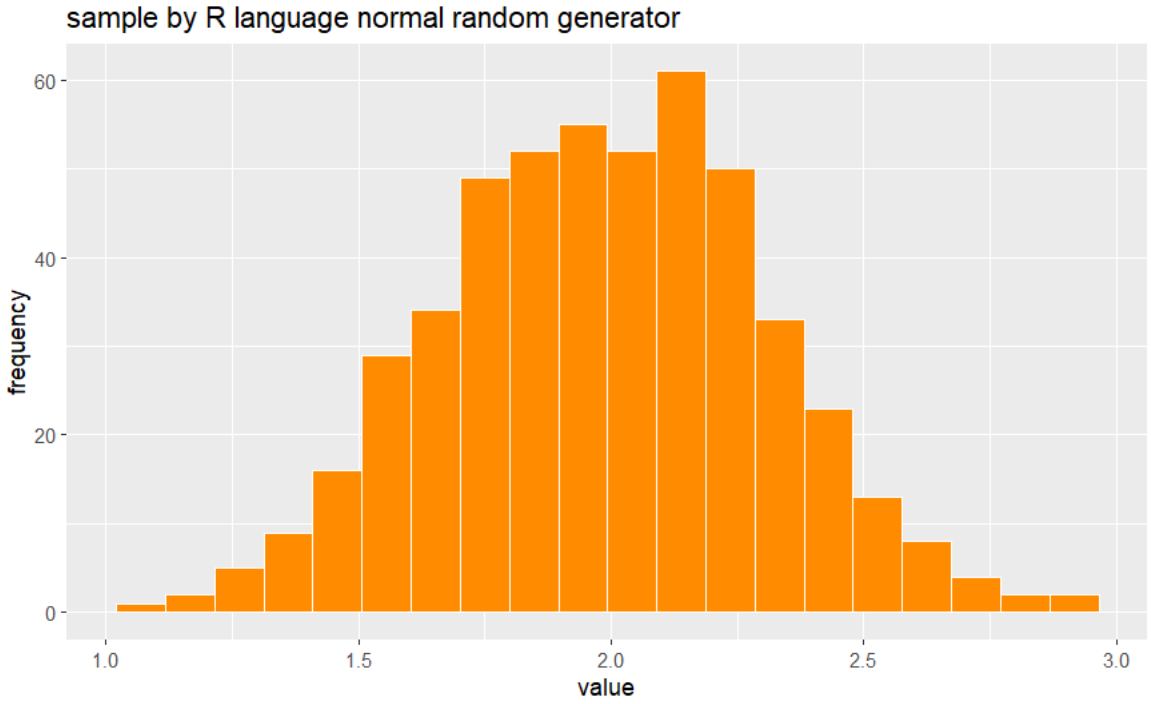

Figure 12. histogram of normal random sample from R

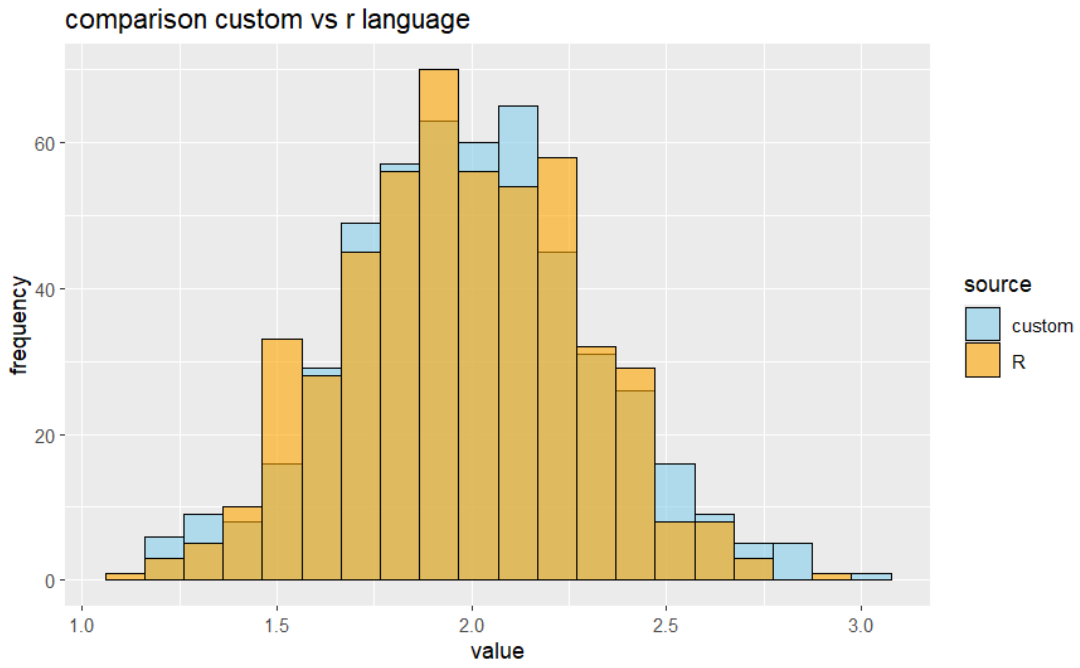

Figure 13. comparision of histograms of normal random sample from python and R

- Kolmogorov-Smirnov test

```
        Asymptotic two-sample Kolmogorov-Smirnov test

data:  norm_emperical and norm_theoretical
D = 0.06, p-value = 0.3291
alternative hypothesis: two-sided
```

Figure 14. result of Kolmogorov-Smirnov test with normal samples with mean 2 and standard deviation 0.3

Conclusion: The p-value is bigger then 0.05 that means that Null hypothesis is successful and it means that the samples are similar and it means that they follow same distribution.

```
        Pearson's Chi-squared test

data:  freq_emperical and freq_theoretical
X-squared = 70, df = 49, p-value = 0.02605
```

- Chi-square test

Figure 15. result of Chi-square test with normal samples with normal samples with mean 2 and standard deviation 0.3

Conclusion: p-value is smaller than 0.05 it seems that the samples is not similar so it means that they may not follow the same distribution.

Testing normal random generator (mean=2.5;sd=0.1)

	Standard Deviation	Mean
Emperical	0.0981424s	2.500823
Theoretical	0.1	2.5

Table 4. emperical end theoretical values for exponential random generator with rate 4

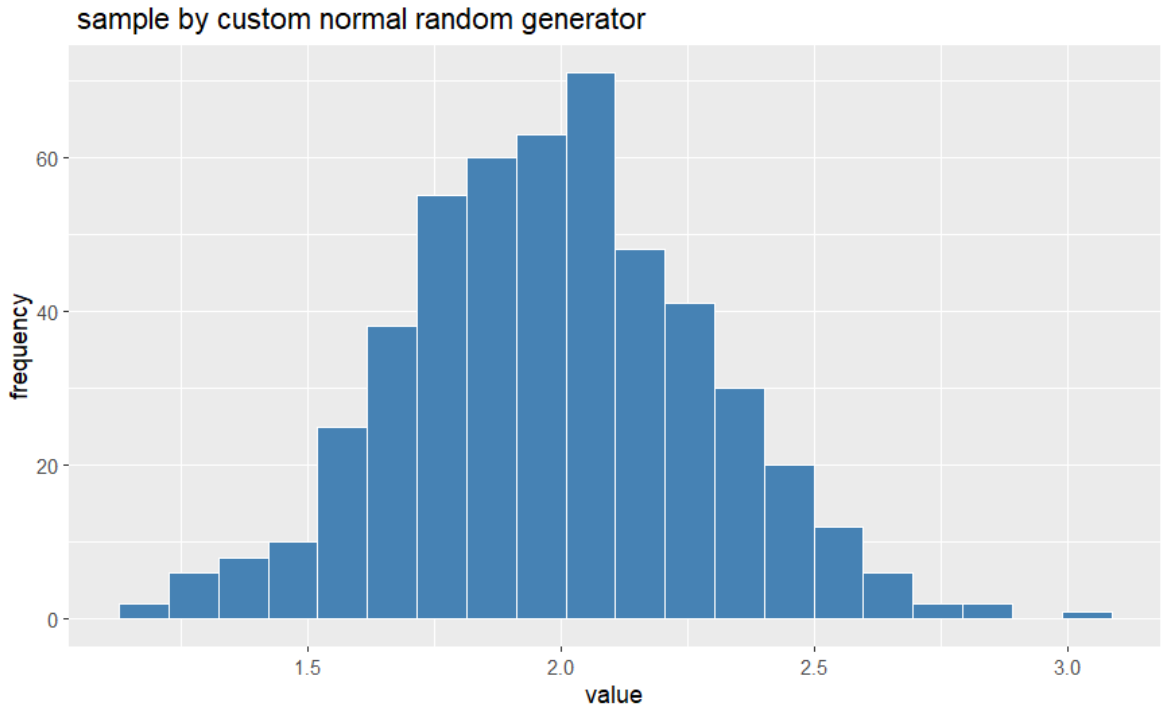

Figure 16. histogram of normal random sample from python

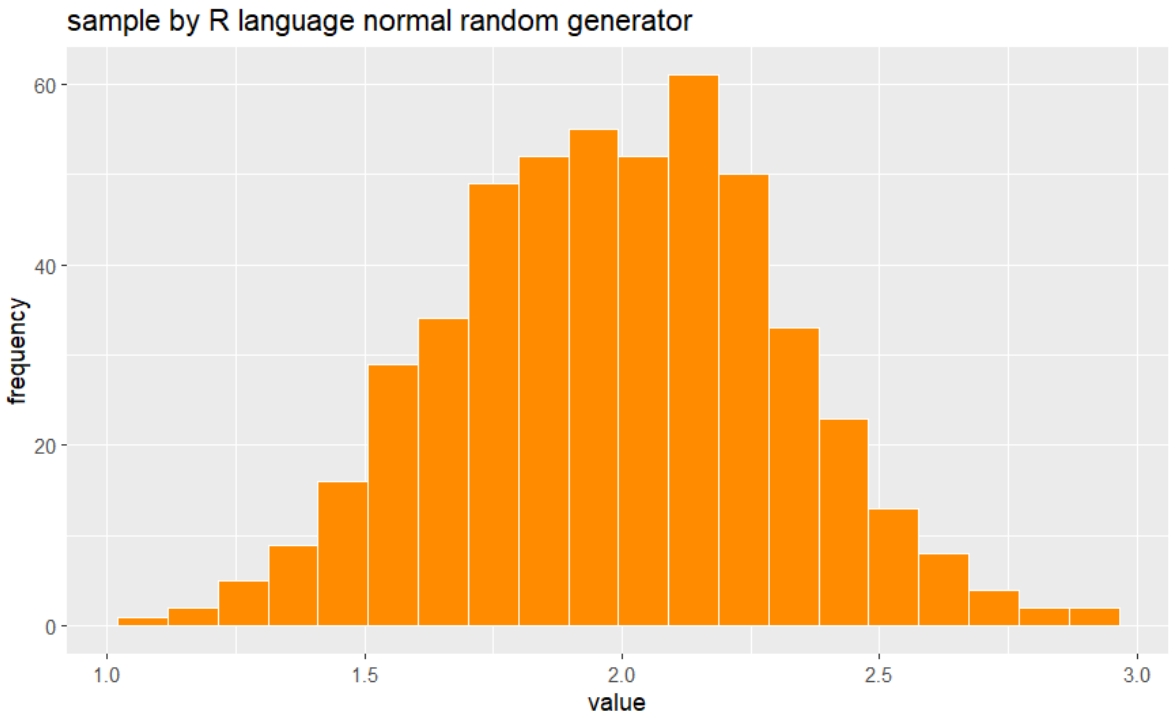

Figure 17. histogram of normal random sample from R

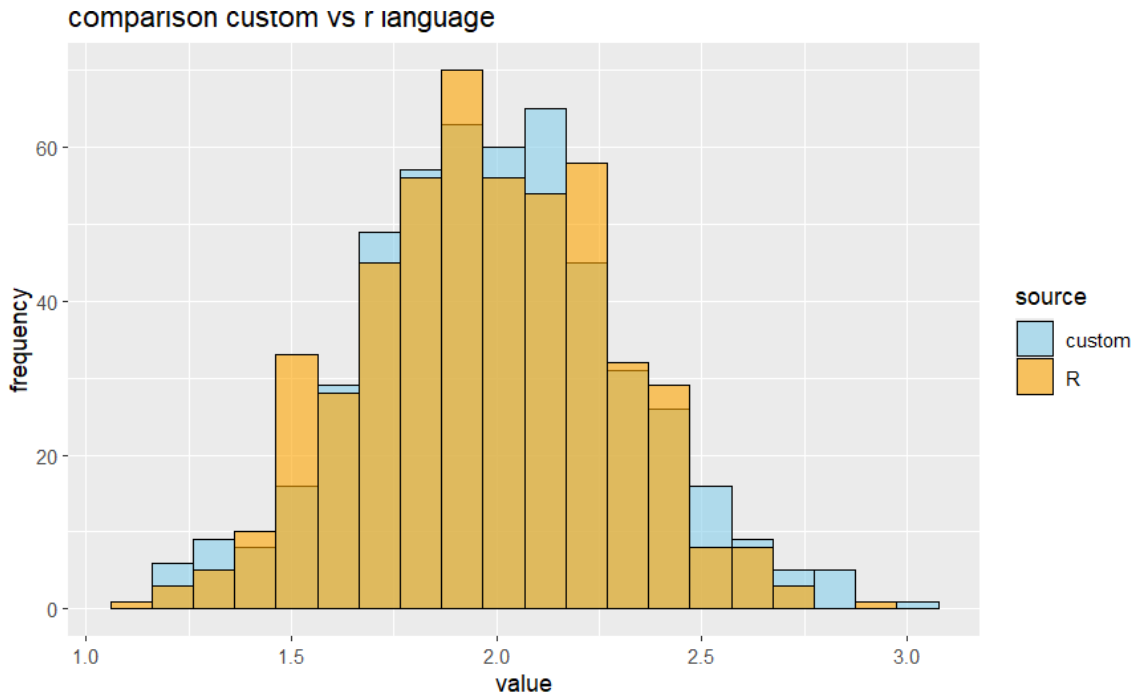

Figure 18. comparision of histograms of normal random sample from python and R

- Kolmogorov-Smirnov test

```
        Asymptotic two-sample Kolmogorov-Smirnov test

data:  norm_emperical and norm_theoretical
D = 0.838, p-value < 2.2e-16
alternative hypothesis: two-sided
```

Figure 19. result of Kolmogorov-Smirnov test with normal samples with mean 2 and standard deviation 0.3

Conclusion: The p-value is smaller then 0.05 it means that the sample is not follow the distribution.

- Chi-square test

```
        Pearson's Chi-squared test

data:  freq_emperical and freq_theoretical
X-squared = 30, df = 21, p-value = 0.09199
```

Figure 20. result of Chi-square test with normal samples with normal samples with mean 2 and standard deviation 0.3

Conclusion: The p-value is bigger then 0.05 that means that Null hypothesis is successful and it means that the samples are similar and it means that they follow same distribution.

Conclusion on statistical results

The all test of sample generated by custom exponential random values generator has success in test and it means that the exponential random value generation mostly works correctly or same as the R language exponential random value generator and it seems that generator works properly. But here is some failed tests by sample generated by the Normal random values generator. In first case the sample have succes in the kolmogorov-smirnov test but it fails the chi-square test . In the second case the kolmogorov-smirnov test is failed but chi-square test is success. It seems that something wrong with custom random value generator or it may have problems in the moment of entering the parameters. It may have incorect position in the call of the method. But the decision is made to use that normal random values generator becaues it have success in the 50% of the test.

Simulation results

The simulation was runed one time and succesfuly finished. Next figure shows the data output. By the output the first and last 20 iterations is collected and provided in the table below.

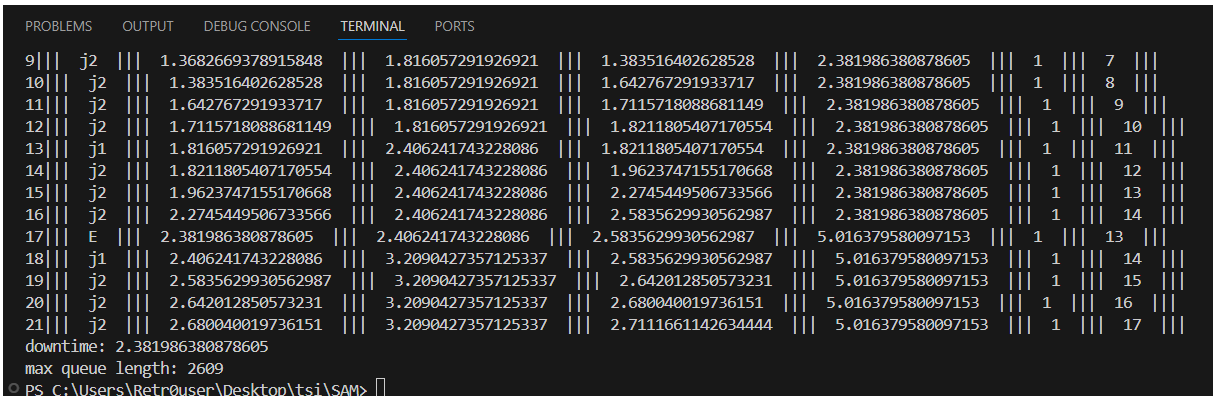

Figure 21. Output of simulation in Python programm

Position	Task	Actual time	J1	J2	Server finish	Server status	Queue size	Queue
0	Start	0	0.59892	0.45410	0	0	0	[]
1	E	0	0.59892	0.45410	500	0	0	[]
2	J2	0.45410	0.59892	0.76429	2.59945	1	0	[]
3	J1	0.59892	1.02035	0.76429	2.59945	1	1	[j1]
4	J2	0.76429	1.02035	0.88257	2.59945	1	2	[j1, j2]
5	J2	0.88257	1.02035	1.43694	2.59945	1	3	[j1, j2, j2]
6	J1	1.02035	2.77529	1.43694	2.59945	1	4	[j1, j2, j2, j1]
7	J2	1.43694	2.77529	1.46575	2.59945	1	5	[j1, j2, j2, j1...]
8	J2	1.46575	2.77529	1.47320	2.59945	1	6	[j1, j2, j2, j1...]
9	J2	1.47320	2.77529	1.73945	2.59945	1	7	[j1, j2, j2, j1...]
10	J2	1.73945	2.77529	1.81461	2.59945	1	8	[j1, j2, j2, j1...]
11	J2	1.81461	2.77529	1.92822	2.59945	1	9	[j1, j2, j2, j1...]
12	J2	1.92822	2.77529	2.43032	2.59945	1	10	[j1, j2, j2, j1...]
13	J2	2.43032	2.77529	2.47331	2.59945	1	11	[j1, j2, j2, j1...]
14	J2	2.47331	2.77529	2.68083	2.59945	1	12	[j1, j2, j2, j1...]
15	E	2.59945	2.77529	2.68083	4.38908	1	11	[j2, j2, j1, j2...]
16	J2	2.68083	2.77529	3.02906	4.38908	1	12	[j2, j2, j1, j2...]
17	J1	2.77529	3.51582	3.02906	4.38908	1	13	[j2, j2, j1, j2...]
18	J2	3.02906	3.51582	3.07778	4.38908	1	14	[j2, j2, j1, j2...]
19	J2	3.07778	3.51582	3.26091	4.38908	1	15	[j2, j2, j1, j2...]
20	J2	3.26091	3.51582	3.43365	4.38908	1	16	[j2, j2, j1, j2...]
...	...	...	...	...	...	...	...	...
N – 19	J1	496.88991	498.28520	497.17811	498.33634	1	2500	[j2, j1, j2, j1...]
N – 18	J2	497.17811	498.28520	497.51207	498.33634	1	2501	[j2, j1, j2, j1...]
N – 17	J2	497.51207	498.28520	497.70354	498.33634	1	2502	[j2, j1, j2, j1...]
N – 16	J2	497.70354	498.28520	497.73777	498.33634	1	2503	[j2, j1, j2, j1...]
N – 15	J2	497.73777	498.28520	497.91737	498.33634	1	2504	[j2, j1, j2, j1...]
N – 14	J2	497.91737	498.28520	497.95061	498.33634	1	2505	[j2, j1, j2, j1...]

N – 13	J2	497.95061	498.28520	498.08563	498.33634	1	2506	[j2, j1, j2, j1...]
N – 12	J2	498.08563	498.28520	498.14349	498.33634	1	2507	[j2, j1, j2, j1...]
N – 11	J2	498.14349	498.28520	498.42829	498.33634	1	2508	[j2, j1, j2, j1...]
N – 10	J1	498.28520	498.50269	498.42829	498.33634	1	2509	[j2, j1, j2, j1...]
N – 9	E	498.33634	498.50269	498.42829	500.76425	1	2508	[j1, j2, j1, j2...]
N – 8	J2	498.42829	498.50269	498.51465	500.76425	1	2509	[j1, j2, j1, j2...]
N – 7	J1	498.50269	498.83558	498.51465	500.76425	1	2510	[j1, j2, j1, j2...]
N – 6	J2	498.51465	498.83558	498.85209	500.76425	1	2511	[j1, j2, j1, j2...]
N – 5	J1	498.83558	500.45761	498.85209	500.76425	1	2512	[j1, j2, j1, j2...]
N – 4	J2	498.85209	500.45761	498.852247	500.76425	1	2513	[j1, j2, j1, j2...]
N – 3	J2	498.852247	500.45761	499.45781	500.76425	1	2514	[j1, j2, j1, j2...]
N – 2	J2	499.45781	500.45761	499.80732	500.76425	1	2515	[j1, j2, j1, j2...]
N – 1	J2	499.80732	500.45761	500.10412	500.76425	1	2516	[j1, j2, j1, j2...]
N – 0	End	500.10412	500.45761	500.10412	500.76425	1	2516	[j1, j2, j1, j2...]

Table 5. Logs from model

Model 2 developing (GPSS)

Code of the model

The second model developed using GPSS modeling language.The algorithm based on moving jobs throw flaged parts.

The job generates in GPSS by opperation Generate this received parameter (in first case the parameter is a value generated by exponential distribution random value generator with parameters of J1).

After arrival time is generate the execution time should be added it happenes by assigning the “time” variable and it receives the generated value by Normal random value generator. The job is placing to the queue names “BUF”. The queue is organized by FIFO(the personal task is the reason of that type of queue usage). The server get the first element from queue to operate with it by command “SEIZE SERVER” and the job removes from the queue by command “DEPART BUF”. Next the value of operating time receives from time by command “ADVANCE P$time”. After, the “server” change the status to free by command “RELESE SERVER” and it exit the simulation by command “TERMINATE”.

The J2 generation is same like the J1 type(genertng arrival time, generating execution time) but it go to the flaged position named “LBL” by command “TRANSFER, LBL”.

Last part of code on GPSS Generates a time limitter by command “GENERATE”. In current case generates the limit that is 500 time units(for simplyfing and better comparision the one time unit repsents as 1 minute). The termination of the simulation happenes by command “TERMINATE 1”

```
GENERATE(Exponential(1,0, 0.67)) ;J1 arrival time
ASSIGN time,(Normal(1, 2, 0.3)) ;processing time for J1

lbl QUEUE BUF ;creation of queue BUF and update the queue

SEIZE SERVER

DEPART BUF

ADVANCE P$time

RELEASE SERVER
TERMINATE 0

GENERATE(Exponential(1,0, 0.25)) ;J2 arrival time
ASSIGN time,(Normal(1, 2.5, 0.5)) ;processing time for J2
TRANSFER ,lbl ; transfer ALL items to lbl

;timer
GENERATE 500
TERMINATE 1

START 1
```

Code block 1. Source code of GPSS model

Simulation results

The simulation happenes by GPSS World student version and provides the result of simulation in the formated page format.

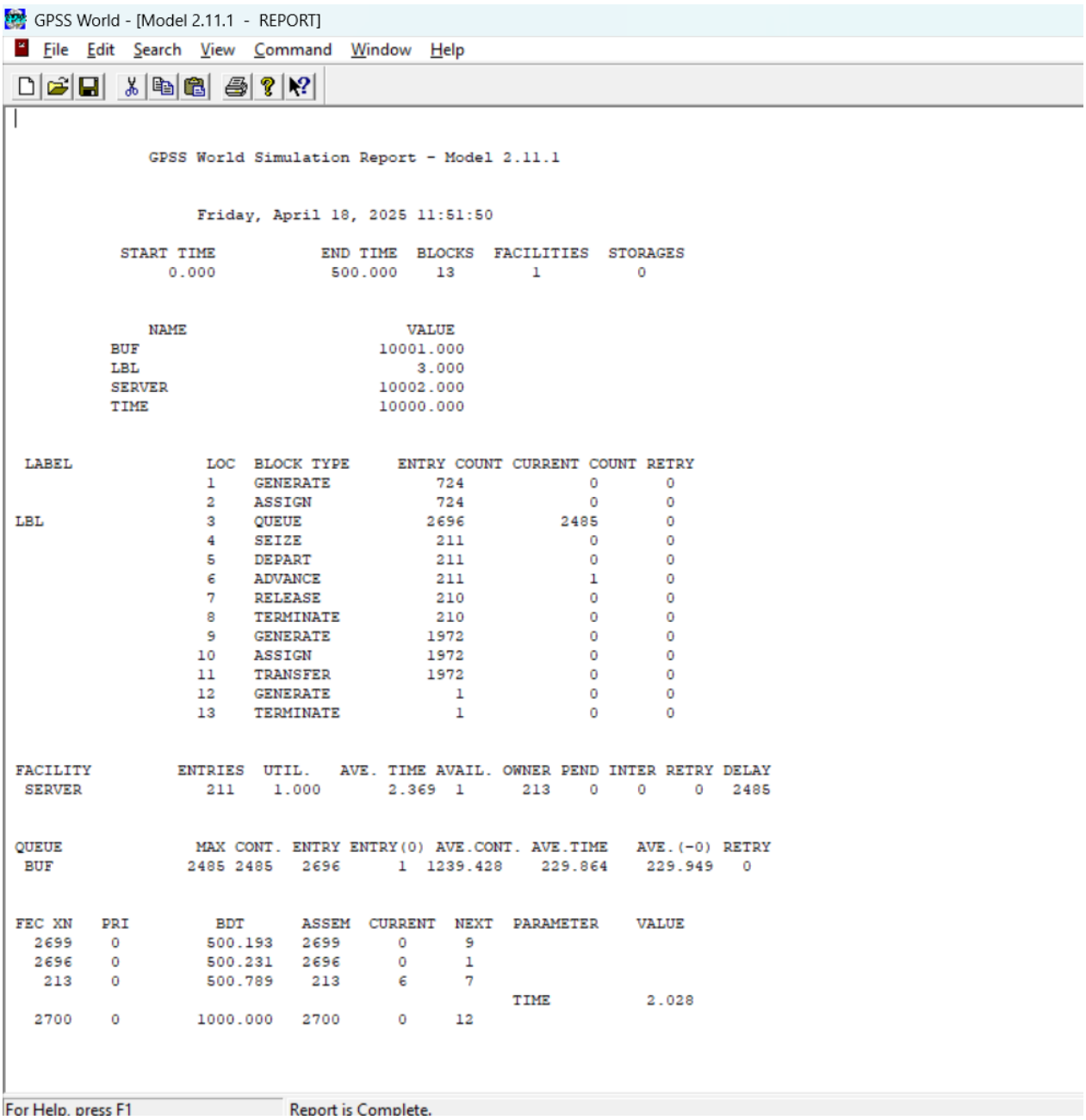

Figure 22. result of GPSS model simulation

Model 3 developing (AnyLogic)

Model overview

That model consist of 4 type of elements(agents):

- Source – entry point that simulate the task apperance and provide it execution time [Q1 and Q2]

- Queue – element that contain the the appered jobs by “source” elements and send next the job from the queue depends to queue type to the next element when it free [Queue]
- Delay – handle the jobs for a specifc time generated previously in the apperance of the job and contains in variable of the object linked to current job.[Server]
- Sink – job break point(the job is finished then it enter that element and disappears, but that element count the amount of entered elements)[sink]

The structure of logic in AnyLogic provided below.

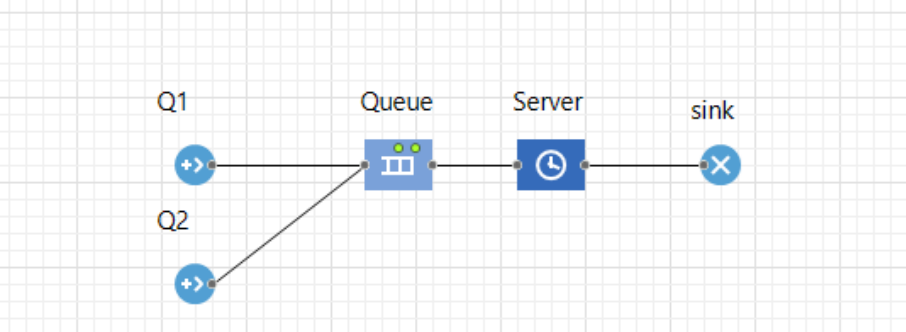

Figure 23. Structure of the model of simulation

The job have 2 types depending assignment task. The 2 source elements are placed to generate 2 types of jobs in same simulation and they receives names Q1 and Q2. According to task the Q1 and Q2 are configured to interarrival time arrivels and generates by exponential random generator with different variable, the full configuration of the elements provided below.

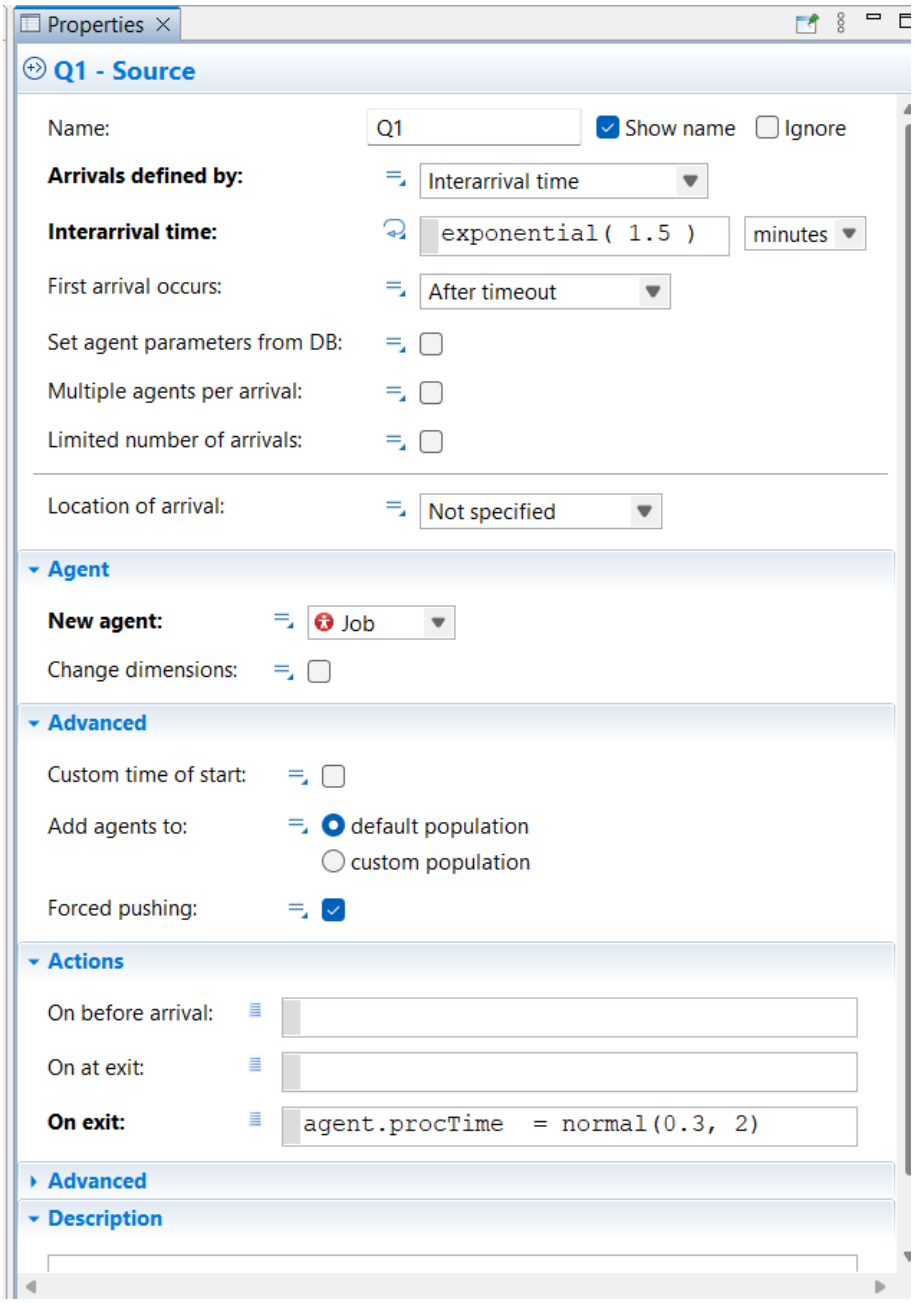

Figure 24. Source Q1 configuration

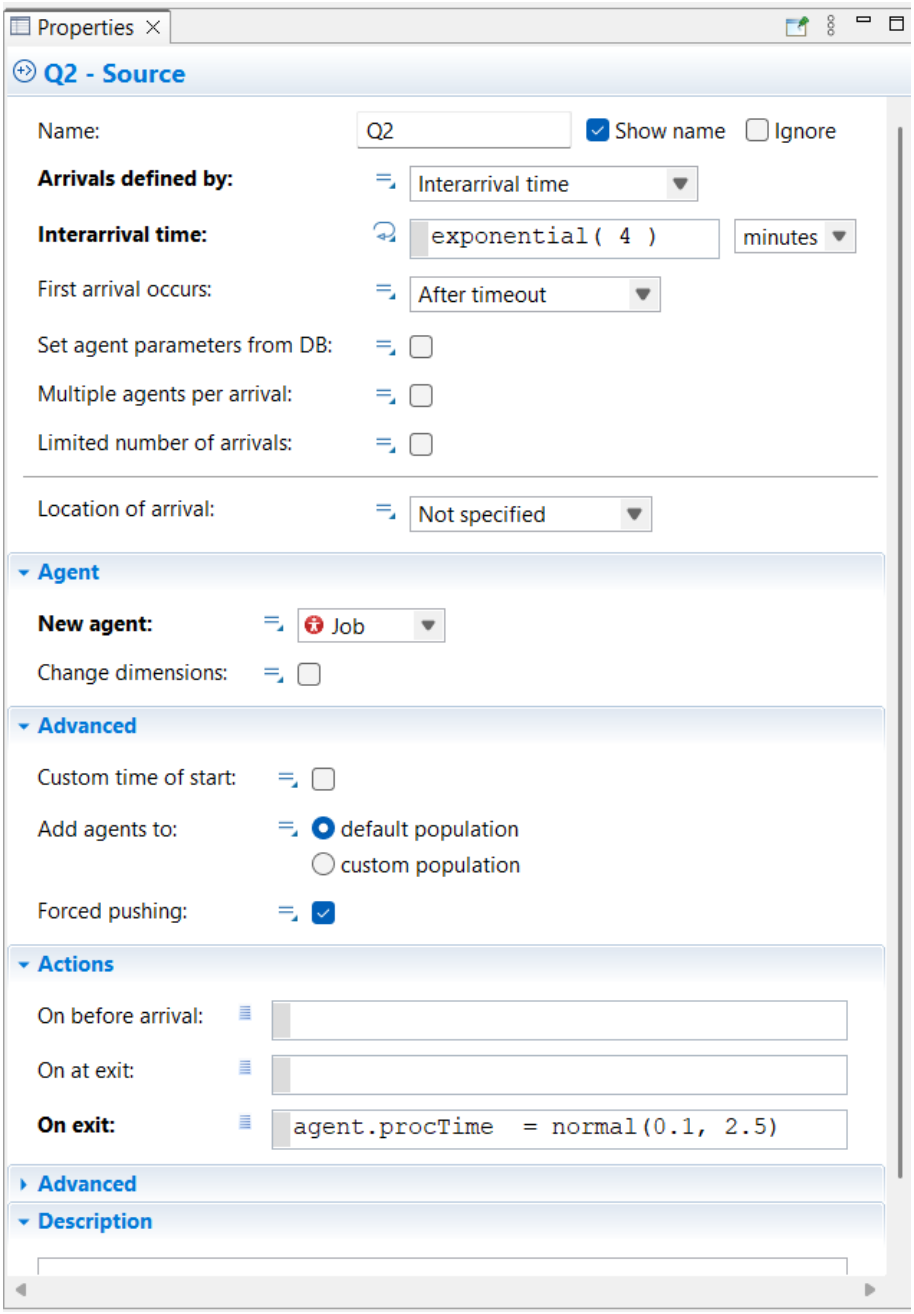

Figure 25. Source Q2 configuration

These source elements generates the agent that names “Job”. These objects have one attribute that contains the time that required for “Job” proccesing on the server. The name of attribute is “procTime” and it receives the value in the “Job” arriving and the value generates by normal random value generator with different parameters, depends to the source or “Job” type.

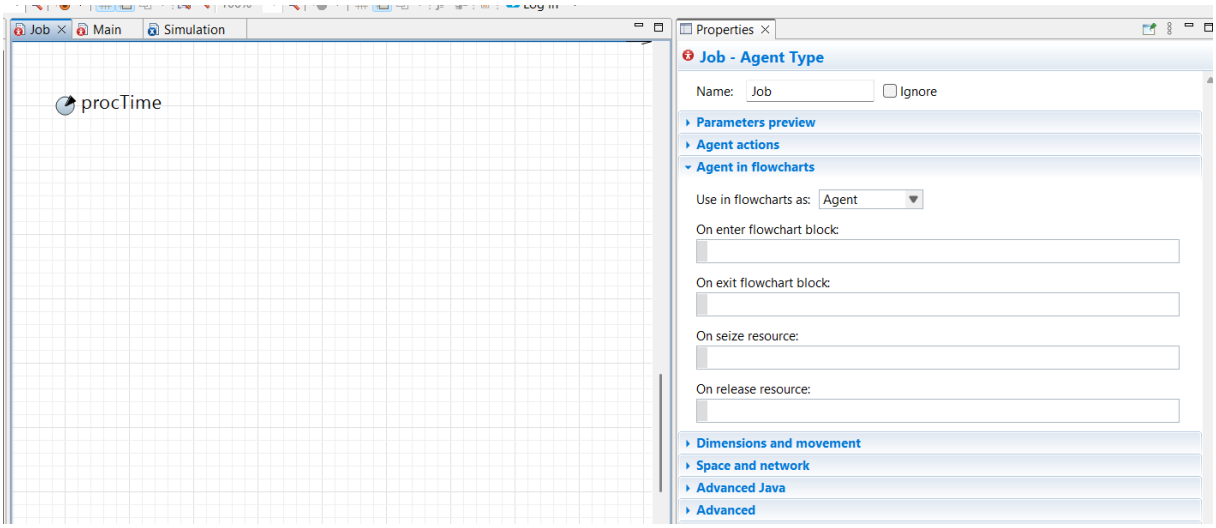

Figure 26. Job class structure

The generated “Jobs” enter the queue that have “FIFO” type. These agents placed in the queue while the server is busy in the moment then server has finished the task then first element from the queue enter server. The decision is made to rise the maximal size of queue from default to 10000 becaues the result previously received by another models shows that maximal size of the queue in the simulations in 2000-3000 and it’s required bigger max size of queue in that model then default(100 elements). The configuration of queue provided below.

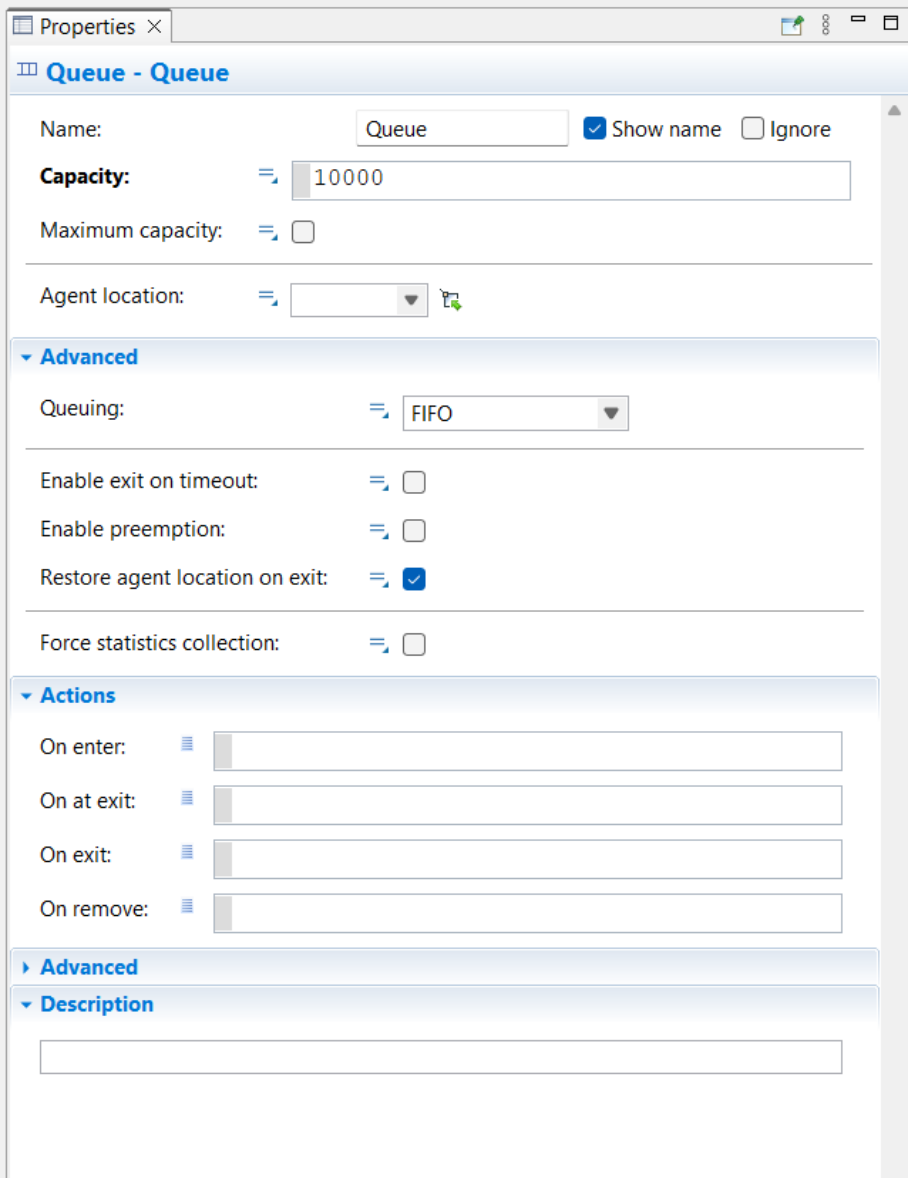

Figure 27. queue configuration in the model

The delay element used as the simulation of server operating because that element delay the object in that position on predefined time. That time is contained in the agent attribute that names "procTime". After the end of delay time the current object enters the next, last element of the model and the object enter the delay section from the queue.

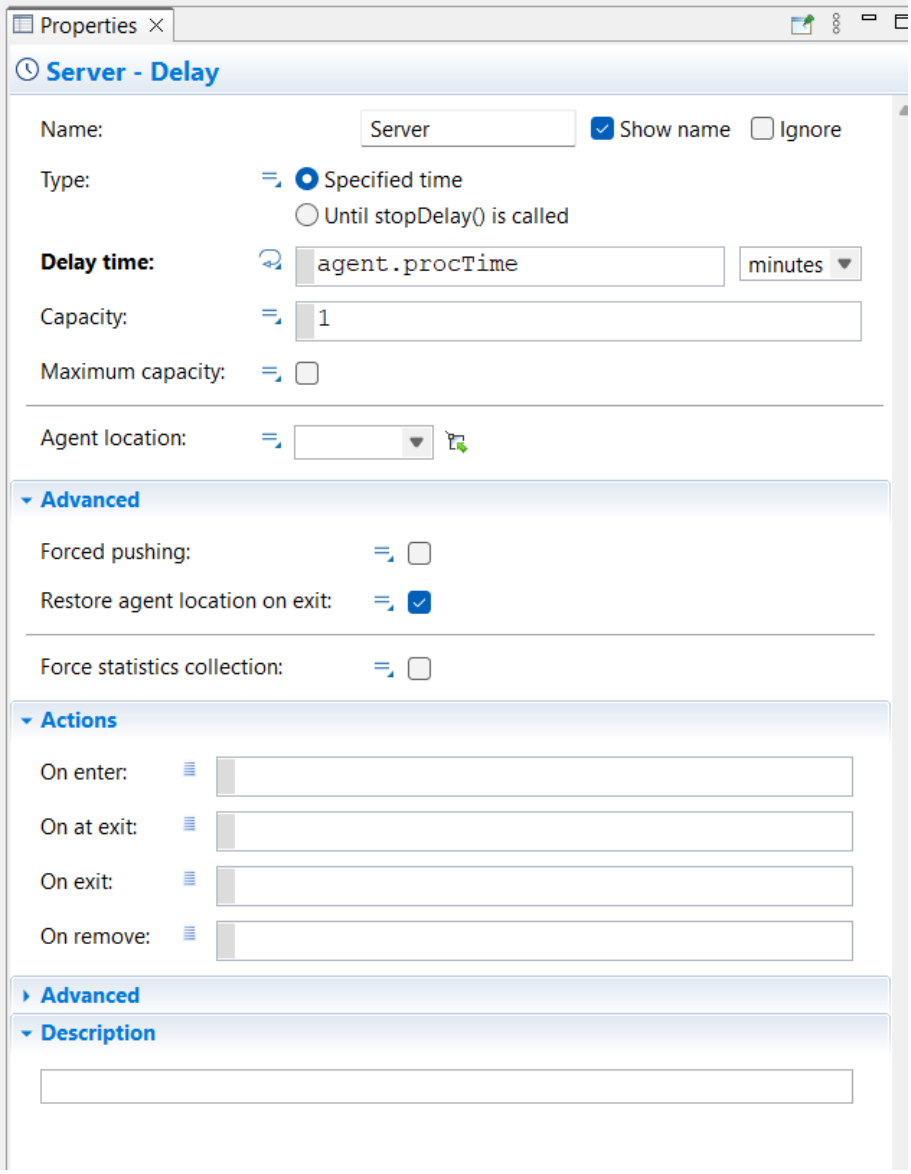

Figure 28. "Server" element configuration

The "Bar Chart" elements is placed and configured to collect MOE values depending on the personal task. These analitical elements connected to the elements of the model depeds to the task. One "Bar Chart" connected to the queue and it shows some information about the queue and amount of object in that queue.

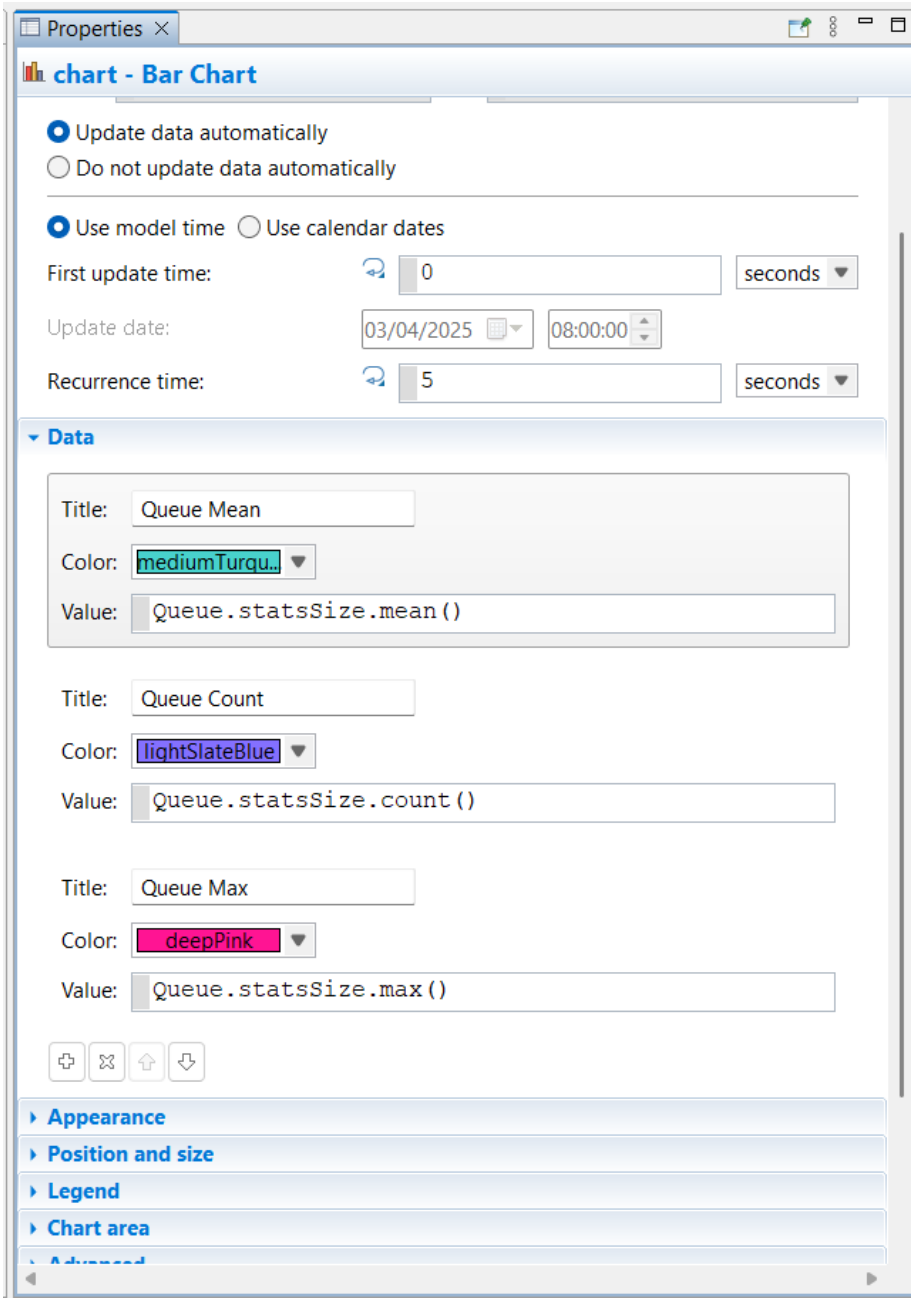

Figure 29. Bar Chart with "Queue" information configuration

The second "Bar Chart" shows the value of server utilization level. The value of delay active time is used as server utilization level, becaues the delay module is used as server in the current model.

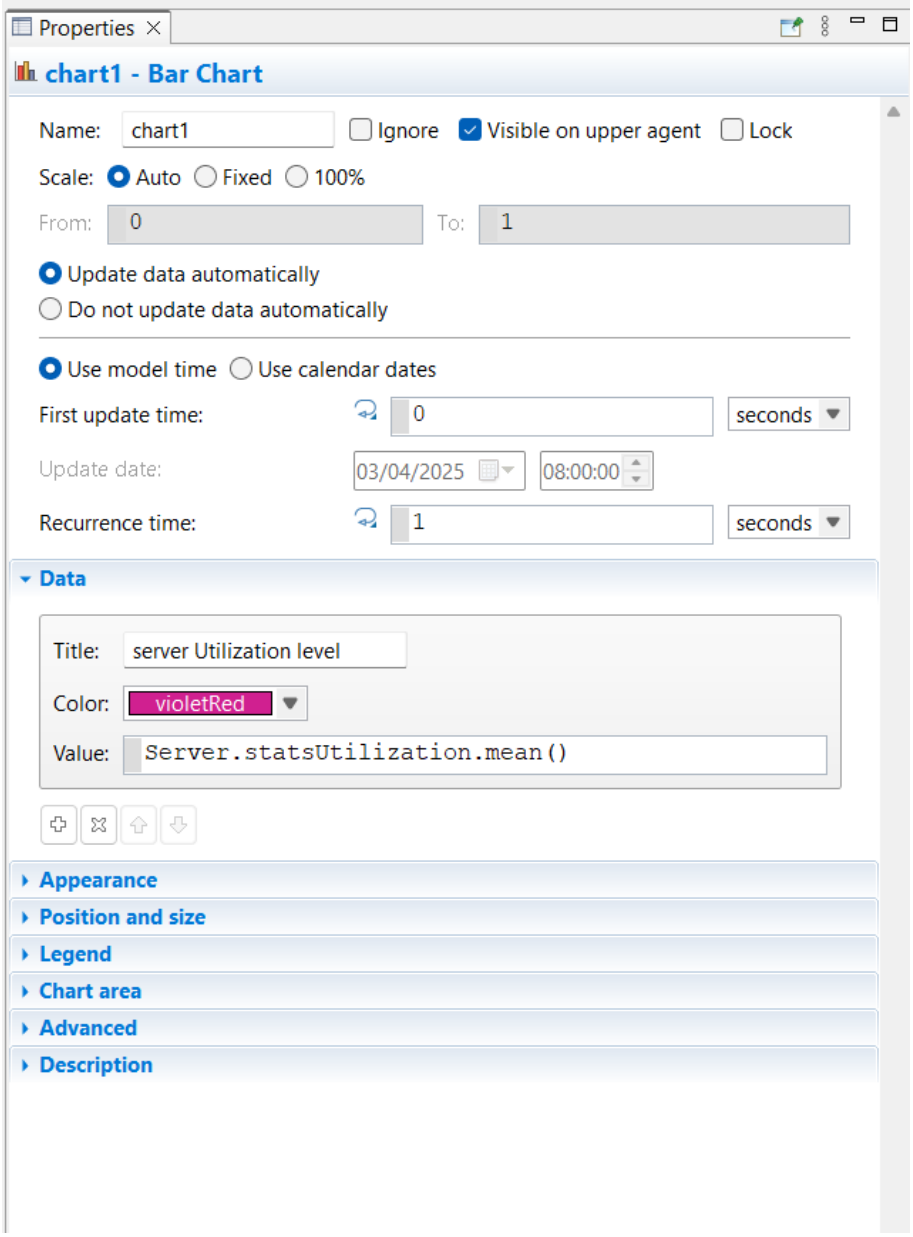

Figure 30. Bar Chart with "Server" information configuration

Simulation results

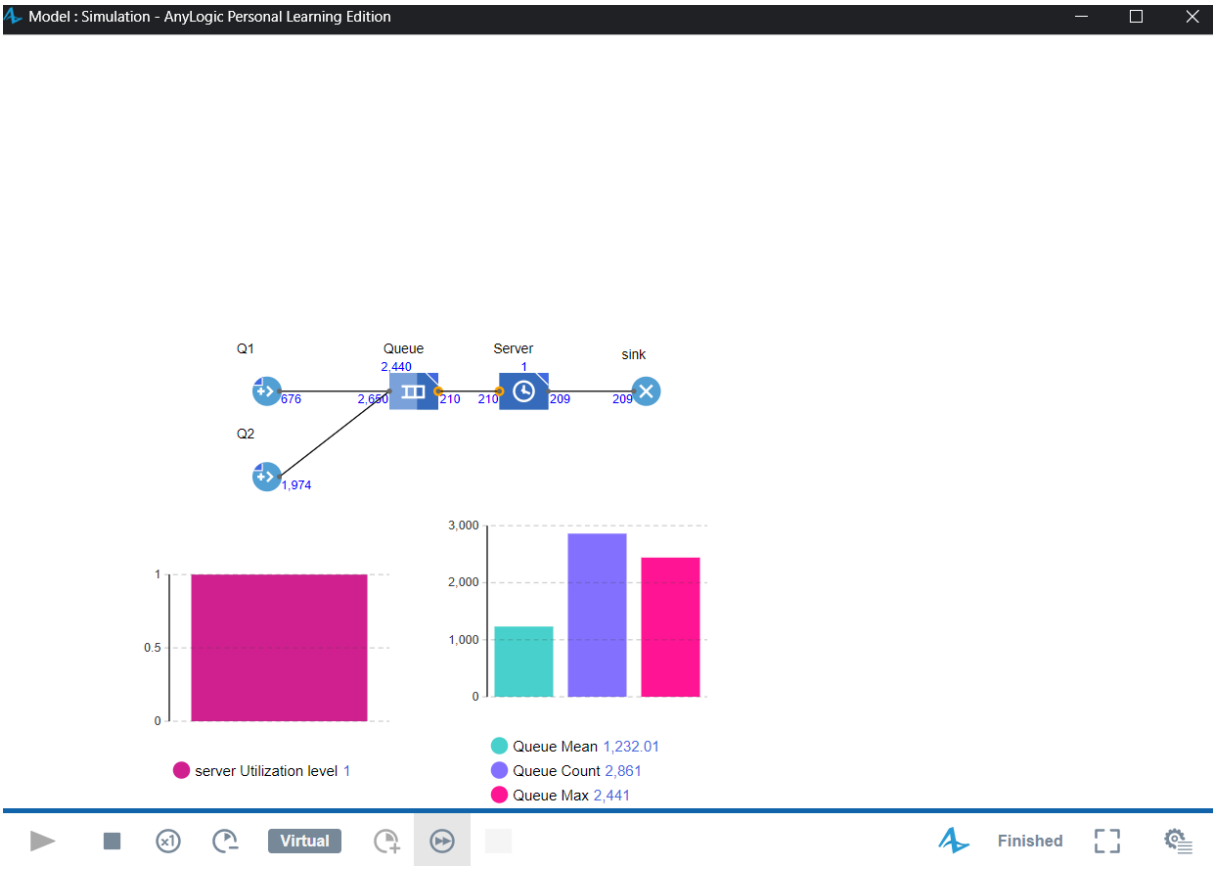

Figure 31. Simulation Result

It seems that the server Utilization level is 1 and it means that the server is bussy for all time of the simulation. But it's required to understand that AnyLogic round the values and the 1 of utilization means that time of dowtime from all time of simulation have realy small part. The figure 32 is provided as proof of that statement. The example of stats of model on 10 minute of simulation provided below.

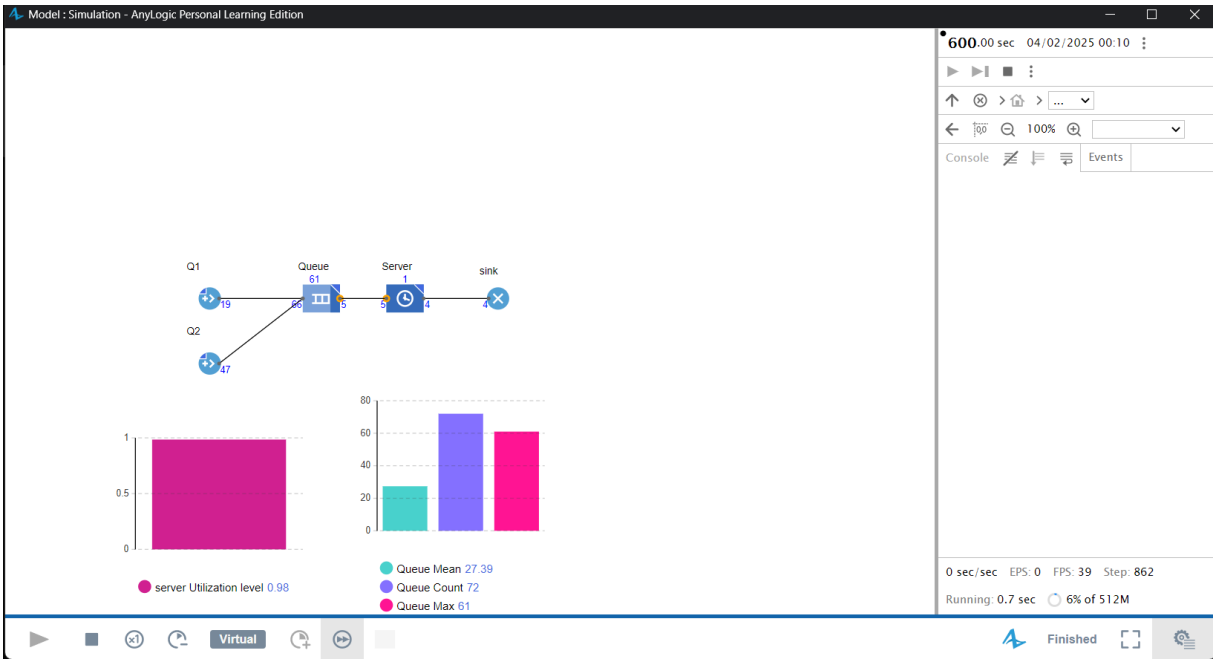

Figure 31. Simulation Result on 10 minute of simulation

Simulation results comparison

Description of comparison

Every model is used for generating 10 simulations of each model to collect the MOE values and compare results of these 3 models. For collecting the different results of the same model it's required to change the seeds on each iteration. The AnyLogic model and Python model have autmated seed change so it's not required additional attention but the GPSS model doesn't have autmated seed change. By that reason it happenes manualy by changing the first parameter of random generator methods. For example, "Normal (1, 2, 0.3)" and in that case the "1" is the seed and it change manualy on each simulation generation.

The result is collected and provided in table and some operations is executed to calculate the specifc statistical values, like mean and median.

Statistical parameters and Naïve approach

Iteration	Model 1		Model 2		Model 3	
	Downtime factor	Max. of all jobs in queue	Downtime factor	Max. of all jobs in queue	Downtime factor	Max. of all jobs in queue
1	2.489341 / 5 = 0.5%	2515	0%	2485	0%	2466
2	1.512657 / 5 = 0.3 %	2397	0%	2497	0%	2526
3	2.229073 / 5 = 0.45%	2535	0.1%	2513	0%	2439
4	2.600646 / 5 = 0.52%	2544	0%	2424	0%	2544
5	2.562217 / 5 = 0.51%	2546	0%	2548	0%	2472
6	2.439048 / 5 = 0.49%	2455	0	2371	0%	2632
7	2.472791 / 5 = 0.49%	2568	0.1%	2545	0%	2555
8	2.411813 / 5 = 0.48%	2555	0%	2450	0%	2557
9	2.602284 / 5 = 0.52%	2543	0%	2510	0%	2508
10	1.944784 / 5 = 0.39%	2625	0%	2514	0%	2582
Stats	Mena: 0.465 Median: 0.49 St. dev: 0.066 95% confidence: 0.424 : 0.506 Range: 0.22	Mena: 2528.3 Median: 2543.5 St. dev: 59.3718 95% confidence: 2491.502 : 2565.098 Range: 228	Mena: 0.02 Median: 0 St. dev: 0.4 95% confidence: -0.005 : 0.0488 Range: 0.1	Mena: 2477.7 Median: 2480 St. dev: 6.6189 95% confidence: 2473.598 : 2481.802 Range: 18	Mena: 0 Median: 0 St. dev: 0 95% confidence: 0 : 0 Range: 0	Mena: 2528.1 Median: 2535 St. dev: 55.5544 95% confidence: 2493.668 : 2562.532 Range: 193

Table 6. Colected data from models

After comparison of statistical values some conclusions can be made:

1. Model 2 (GPSS) and model 3 (AnyLogic) have closest downtime factor
2. Model 1 (Python) and model 3 (AnyLogic) have closes mean and median of max jobs in queue

3. Model 2(GPSS) and model 3(AnyLogic) have rounding and it changes the final result of downtime factor

Test hypothesis for homogeneity

Hypothesis for tests

- T-test (Student T test)

Null hypothesis: the means of downtime samples are same

Alternative: The means of downtime samples are different, and all coincidences are accidental

- U-test (Wilcoxon rank sum test with continuity correction)

Null hypothesis: The median difference between the two downtime samples is zero.

Alternative: The median difference between the two downtime samples is not close enough to zero

Test results and conclusions

Models' comparison	T-test result	U-test result	Conclusion
M1 & M2	T = 18.776 df = 12.545 p-value < 0.01 95% conf interval: 0.4024 – 0.5075	W = 100 p-value < 0.01	Null hypothesis is rejected and the downtime factors of model 1 and model 2 are different and don't have real same elements
M1 & M3	T = 21.066 df = 9 p-value < 0.01 95% conf interval: 0.4151 – 0.5149	W = 100 p-value < 0.01	Null hypothesis is rejected and the downtime factors of model 1 and model 3 are different and don't have real same elements
M2 & M3	T = 1 df = 9 p-value = 0.3434 95% conf interval: -0.0126 – 0.0326	W = 55 p-value = 0.3681	Null hypothesis is accepted, and the downtime factor is same for model 2 and model 3

Table 7. Results of the downtime factor samples tests

Models' comparison	T-test result	U-test result	Conclusion
M1 & M2	T = 1.6082 df = 17.759 p-value = 0.1254 95% conf interval: -13.10548 – 98.30548	W = 75 p-value = 0.06	Null hypothesis is success and the max size of queue of model 1 and model 2 are the same with small difference
M1 & M3	T = 0.0073792 df = 17.921 p-value =0.9942 95% conf interval: -56.7599 – 57.1599	W = 50 p-value = 1	Null hypothesis is success and the max size of queue of model 1 and model 3 are the same with small difference
M2 & M3	T = -1.6593 df = 17.954 p-value = 0.1144 95% conf interval: -96.09326 – 11.29326	W = 31 p-value = 0.1655	Null hypothesis is accepted, and the max size of queue of model 2 and model 3 are the same with small difference

Table 8. Results of the max in queue samples tests

Conclusion

That assignment requires to develop 3 models with different ways and tools and compare it's pross and cons. By that reason these 3 models should use different tools and it should deeply described how it works and how it developed.

It seems that the development of the first model by Python language is longer then another tools but it's seems that the Python developed model gives more accurate results of statistical variables like server activity. The model 1 is too different from another 2 models. The main problem of it may be a representation of calculation the downtime factor. In the Python code it calculates by collecting the time of apperance of first appered element in the moment of entering server by the first job. Model prepared for cases then the model have a downtime not in the start of simulation but in current case the downtime has a place in the start of simulation only and all another time it's busy and big queue of the task. It can be seem by max amont of elements in the queue, it's bigger then 2000 elements in the 500 minute of the simulation.

The down time factor rounds in second(GPSS) and third(AnyLogic) model. By that reason in most cases the result of simulation have downtime that equal 0% of all time. By that reason it's realy hard to compare in current case, in case then the downtime is realy small. But the amount of the max tasks in the queue seems close so the models generates results that is too close.

Actualy the AnyLogic is easiest way to create a simulation. It has user friendly interface giant amount of configuration for modeling and big library of pre-defined objects and agents. But it's require to understand that AnyLogic is not free for usage and it's required to get specific subscriptions.

The GPSS seems realy old language specified for simulation and model development. But it's not realy comfortable to use. The GPSS is the best option if the AnyLogic is not accesible and Python is unknown language for developer, because GPSS language have specific structure that makes easer to construct the logic of the simulation model in more simple format.

Source of code

The all developed code and models is placed on GitHub in current repository: