VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

System Performance Evaluation

Assignment 1

The Queuing Systems-based Performance Evaluation Project

Advisor(s):	Nguyen Van Minh Man	
	Nguyen Phuong Duy	
Student(s):	Ho Thanh Binh	ID 2010929
	Nguyen Trong Duy	ID 1852296
	Do Nguyen Huy Hieu	ID 2053003
	Nguyen Hoang Thuan	ID 2052729
	Nguyen Nhat Tien	ID 2053496
	Dai Ngoc Quoc Trung	ID 2053537

HO CHI MINH CITY, MAY 2024

Contents

List of Figures

1 Member list & Workload

No.	Full name	Student ID	Contribution
1	Do Nguyen Huy Hieu	2053003	100%
2	Nguyen Trong Duy	1852296	100%
3	Nguyen Hoang Thuan	2052729	100%
4	Ho Thanh Binh	2010929	100%
5	Nguyen Nhat Tien	2053496	100%
6	Dai Ngoc Quoc Trung	2053537	100%

2 Introduction

Topic: Write a simulation of a phone teller system employing multiple departments, i.e., customer service, product guarantee team, and technical support. For more variants of system settings, you can employ the up-to-date robotics/AI-enabled teller to increase the system capacity.

Reason for choosing topic:
In the ever-evolving field of computer science, the study and application of queuing systems have emerged as a critical area of focus. This report is centered on Topic 8, which explores the implementation of various types of single and multiple queuing systems. The choice of this topic was motivated by its practical relevance and the opportunity it presents to apply theoretical concepts to real-world scenarios.

By using Kendall's notation and the SimPY library, we aim to build a comprehensive application that can accurately model and solve complex problems. This project not only deepens our understanding of queuing theory but also provides valuable experience in computational problem-solving. The following sections will detail our methodology, design choices, and evaluation metrics, offering a complete overview of our journey from concept to implementation.

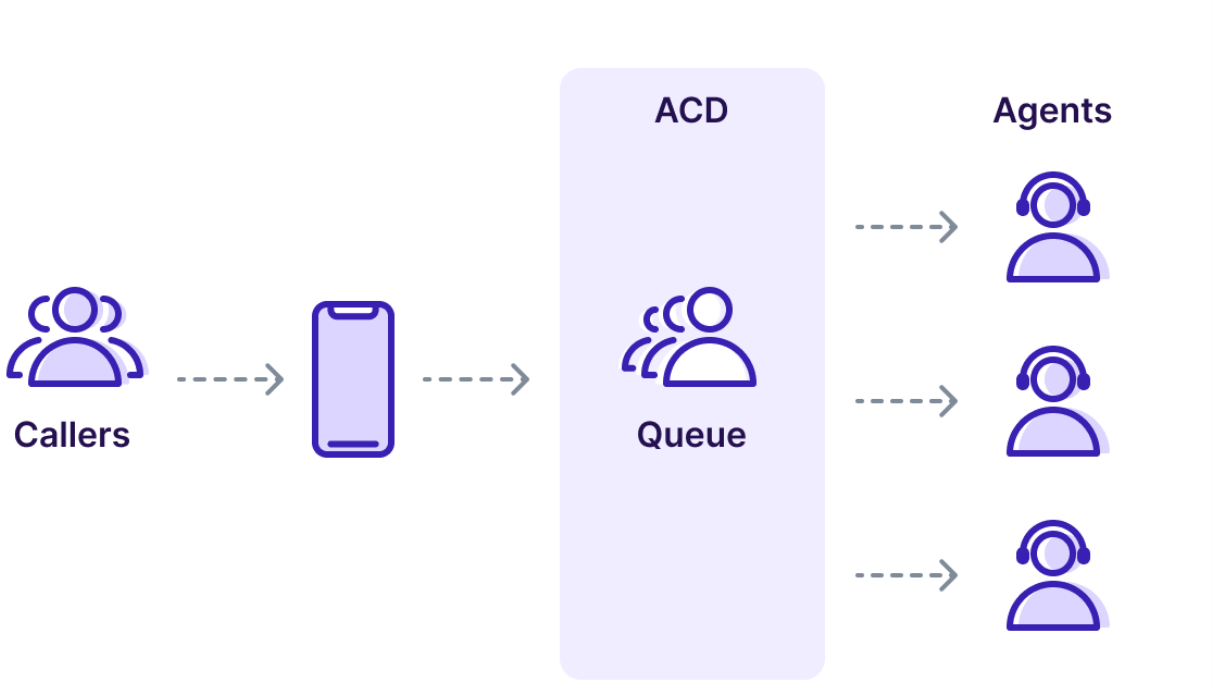

Figure 2.1: Overview system

3 System design

3.1 Define Goal and System

- Objective: To assess the effectiveness of a phone system that utilizes multiple departments (customer service, product guarantee team, and technical support) and potentially an AI-assisted teller.
- Purpose: The system's function is to direct incoming calls to the appropriate department, depending on the customer's needs.

  To apply a queuing system to the phone teller system. Here's how you can map the components:

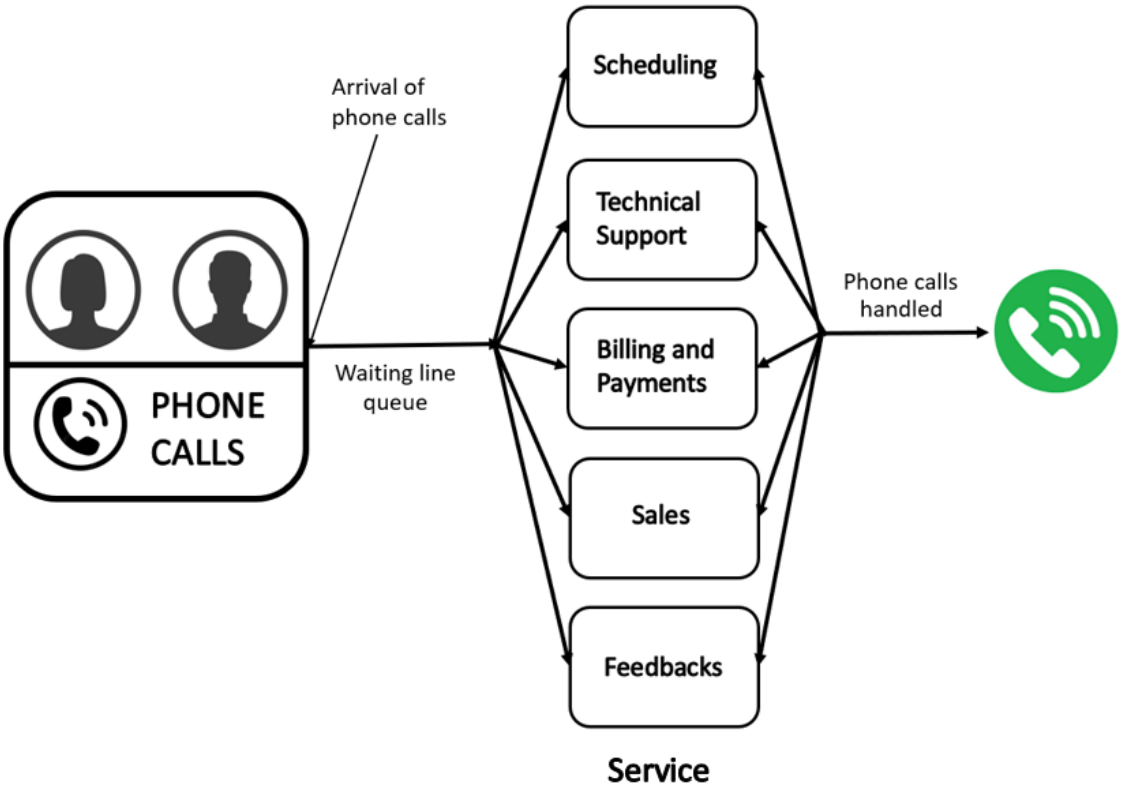

Figure 3.1: Phone call system

3.2 Service and Outcomes

- Scheduling:
  Services: Let the customer choose their desired service and direct the call to the corresponding department. Outcomes:
  - Correct: Customer finds the suitable option.
  - Incorrect: Customer chooses other options.
  - Unavailable: Customer can not connect to the call.
- Customer service:
  Services: Answering questions about products, resolving complaints, following up on customer issues, and providing guidance and advice for a positive customer experience. Outcomes:
  - Correct: Customer satisfied with the inquiry.
  - Incorrect: Customer is not happy with the inquiry.
  - Unavailable: Customer can not connect to the call.
- Technical support: Services: Assisting customers with technical issues, providing guidance on how to fix problems, and conducting repairs and replacements as needed. Outcomes:
  - Correct: Customer successfully tackles the issue.
  - Incorrect: Customer is not able to fix it.
  - Unavailable: Customer can not connect to the call.
- Product guarantee:
  Services: Offering a full refund or a free replacement of the product. Outcomes:
  - Correct: Customer is willing to the suggestion.
  - Incorrect: Customer is not satisfied with the suggestion.
  - Unavailable: Customer can not connect to the call.
- Sales:
  Services: Providing information about products and services, handling new orders, and supporting upselling or cross-selling activities. Outcomes:

  - Correct: Customer is willing to buy the product.
  - Incorrect: Customer refuses to buy that product.
  - Unavailable: Customer is unable to connect to a call.

- Feedback:
  Services: Accumulating customer experience while using the product for improvement and upgrading. Outcomes:
  - Correct: Customer shares the experience.
  - Incorrect: Customer refuses to share their thoughts.
  - Unavailable: Customer is unable to connect to a call.

3.3 Select Metrics

- Scheduling:
  1. Average scheduling time: The average time it takes to schedule an appointment.
  2. Accuracy: The proportion of appointments that start on time.
  3. Rescheduling rate: The proportion of appointments that need to be rescheduled.
- Technical support:
  1. First call resolution rate: The percentage of technical issues resolved during the first call.
  2. Average handle time: The average time it takes to resolve a technical issue.
  3. Customer satisfaction: This could be measured through post-call surveys.
- Product guarantee:
  1. First call resolution rate: The probability of duplicate calls.
  2. Average handle time: The mean duration of a customer interaction with the representative.
  3. Experience rating: It relates to the customer's experience during the service.
- Sales:
  1. Conversion rate: The percentage of inquiries that result in a sale.
  2. Cross-sell and upsell rate: The percentage of sales calls where an additional product or service is sold.
  3. Average order value: The average value of the orders placed.
- Feedback:
  1. Feedback response rate: The percentage of customers who provide feedback when inquired.
  2. Positive feedback rate: The percentage of feedback responses that are positive.
  3. Implementation rate: The percentage of customer suggestions that are implemented.

- Additional metrics:

    1. Average call waiting time: The mean time that a customer waits in a queue before servicing.
    2. Probability of abandoned calls: The percentage of unreachable calls.
    3. Probability of calls handled by robotics/AI-enabled mechanism.
    4. Probability of calls resolved by robotics/AI-enabled mechanism.
    5. Average call handling time for robotics/AI-enabled mechanism.
    6. Customer satisfaction rating for robotics/AI-enabled mechanism.

3.4 List System Parameters

- λ: arrival rate. The average number of entities per unit of time.
- μ: service rate. The average number of services per unit of time.
- C: number of servers. It depends on each department in the system.
- ρ: utilization or traffic intensity. This is the arrival-to-service ratio and it indicates how busy the system is.
- Number of calls handled by robotics/AI-enabled tellers.
- Number of robotics/AI-enabled tellers.

3.5 List Factors to Study

- Arrival Rate (λ): This is the average number of calls arriving per unit of time. The levels could be "Low", "Medium", and "High" corresponding to few, moderate, and many calls respectively.
- Service Rate (μ): This is the average number of calls a teller (human or AI) can handle per unit of time. The levels could be "Slow", "Average", and "Fast" corresponding to the speed at which calls are handled.
- Number of Servers (C): This refers to the number of tellers available in each department. The levels could be based on the actual number of tellers, such as "Few", "Some", and "Many".
- Utilization or Traffic Intensity (ρ): This is the ratio of arrival rate to service rate and indicates how busy the system is. The levels could be "Under-utilized", "Optimally-utilized", and "Over-utilized".
- Number of Calls Handled by AI-enabled Tellers: This is the number of calls handled by AI-enabled tellers. The levels could be "Few", "Some", and "Many".
- Number of AI-enabled Tellers: This is the total number of AI-enabled tellers in the system. The levels could be "Few", "Some", and "Many".

4 System Implementation

4.1 Mathematical Approach

4.1.1 Theory

We decided to develop the system with multiple queues of M/M/C/K. Therefore, it is important to display which equations and calculations we chose to work on.

- Steady state system size probabilities:

$$p_n = \begin{cases} \frac{\lambda^n}{n!\mu^n} p_0, & 0 \leqslant n < c \\ \frac{\lambda^n}{c^{n-c} c!\mu^n} p_0, & c \leqslant n \leqslant K \end{cases} \tag{4.1}$$

Since $\sum_{n=0}^{K} p_n = 1$, we get $p_0 = \left(\sum_{n=0}^{c-1} \frac{\lambda^n}{n!\mu^n} + \sum_{n=c}^{K} \frac{\lambda^n}{c^{n-c} c!\mu^n} \right)^{-1}$

$$\Longrightarrow p_0 = \begin{cases} \left[\frac{r^c}{c!} \left(\frac{1-\rho^{K-c+1}}{1-\rho} \right) + \sum_{n=0}^{c-1} \frac{r^n}{n!} \right]^{-1}, & \rho \neq 1 \\ \left[\frac{r^c}{c!} (K-c+1) + \sum_{n=0}^{c-1} \frac{r^n}{n!} \right]^{-1}, & \rho = 1 \end{cases} \tag{4.2}$$

$\left(With\ r = \frac{\lambda}{\mu}, \rho = \frac{r}{c} \right)$

- Expected queue length (L_q) for $\rho \neq 1$:

$$\begin{aligned} L_q &= \sum_{n=c+1}^{K} (n-c) p_n = \frac{p_0 r^c}{c!} \sum_{n=c+1}^{K} (n-c) \frac{r^{n-c}}{c^{n-c}} \\ &= \frac{p_0 r^c \rho}{c!} \sum_{i=1}^{K-c} i\rho^{i-1} = \frac{p_0 r^c \rho}{c!} \frac{d}{d\rho} \left(\sum_{i=0}^{K-c} \rho^i \right) = \frac{p_0 r^c \rho}{c!} \frac{d}{dp} \left(\frac{1-\rho^{K-c+1}}{1-\rho} \right) \\ &= \frac{p_0 r^c \rho}{c!(1-\rho)^2} \left[1 - \rho^{K-c+1} - (1-\rho)(K-c+1)\rho^{K-c} \right] \end{aligned} \tag{4.3}$$

For $\rho = 1, L_q$ can be obtained similarly.

- For this finite capacity waiting space model, $L = L_q + r$ is not true, since a fraction p_K of arrivals do not join the system (when there is no waiting space).

- Effective arrival rate seen by the servers is $\lambda_{eff} = \lambda(1-p_K)$ by (PASTA property).

- The relation between L (the expected system length) and L_q for this model is modified as

$$L = L_q + \frac{\lambda_{eff}}{\mu} = L_q + \frac{\lambda(1-p_K)}{\mu} = L_q + r(1-p_K) \tag{4.4}$$

- By Little's law, the expected waiting time:

$$\begin{aligned} W &= \frac{L}{\lambda_{eff}} = \frac{L}{\lambda(1-p_K)} \\ W_q &= W - \frac{1}{\mu} = \frac{L_q}{\lambda_{eff}} \end{aligned} \tag{4.5}$$

- In such finite capacity queues, an important performance measure is p_K, the probability of blocking (or the proportion of arrivals turned away as they could not enter the system).

4.1.2 Result

- Node 1:

$$\begin{cases} \lambda = 50, & \mu = 50 \\ c = 3, & K = 60 \end{cases}$$

$$\Longrightarrow \begin{cases} \textit{The expected waiting time } E(W) & \approx 0.0209(mins) \\ \textit{The expected servicing time } E(S) & \approx 0.0067(mins) \end{cases}$$

- Node 2:
$$\begin{cases} \lambda = 50 * 0.7 * 3 + 40 * 3 * (1/3), & \mu = 60 * 3 \\ c = 3, & K = 70 \end{cases}$$
$$\implies \begin{cases} E(W) & \approx 0.0354(mins) \\ E(S) & \approx 0.0056(mins) \end{cases}$$

- Node 3, 4, 5 (Multiple Nodes):
$$\begin{cases} \lambda = 60 * 3, & \mu = 40 \\ c = 3 * 3, & K = 40 * 3 \end{cases}$$
$$\implies \begin{cases} E(W) & \approx 0.0253(mins) \\ E(S) & \approx 0.0028(mins) \end{cases}$$

4.1.3 Comparison:

Simulation and mathematical modeling are two essential techniques used in system performance evaluation. This study compares and analyzes the strengths, limitations, and practical implications of these two methods with the aim of providing insights into the most suitable technique for real-world problem-solving scenarios.

- In Node 1 and Node 2, the simulation results are fairly close to the mathematical results, with few differences. This suggests that the simulation model is providing reasonable approximations of the expected service and waiting times.

- In Node 3, 4, and 5 (the multi-queuing node), the simulation results and mathematical results differ, with the simulation producing smaller values for both expected service and waiting times. This could be due to the specific characteristics of the simulation, including the random nature of customer arrivals and service times, which can result in variations.

Advantages: Simulation is more flexible and adaptable when it comes to analyzing different scenarios and assessing the system's performance with heavy load data. It also provides the result faster than the mathematical approach.

Drawbacks:

- Simulation of complex models or long runs can be computationally intensive, requiring significant time and computing resources. Mathematical models, on the other hand, provide instantaneous results.

- Validating simulation models can be challenging and time-consuming, requiring extensive data. Mathematical models are easier to validate as they rely on established equations and assumptions.
- To craft an accurate simulation, one often needs detailed data about various aspects of the system—like service time distributions, arrival patterns, and resource constraints. In the absence of this data, assumptions have to be made, which can introduce errors.

4.2 Simulation Approach

4.2.1 System Architecture:

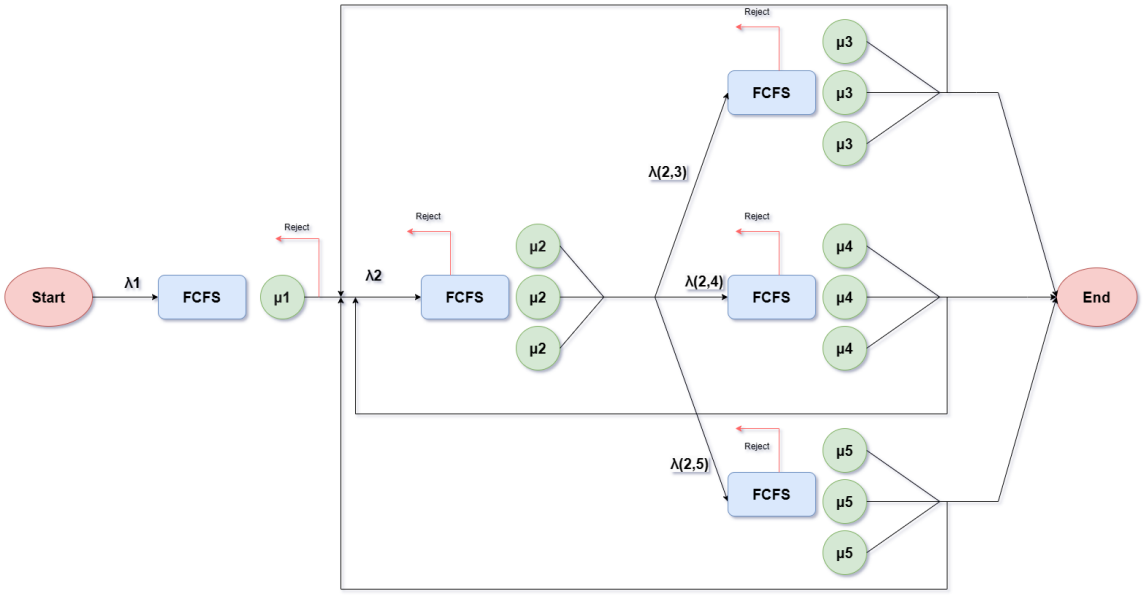

Figure 4.1: System Architecture

4.2.2 Class Diagram:

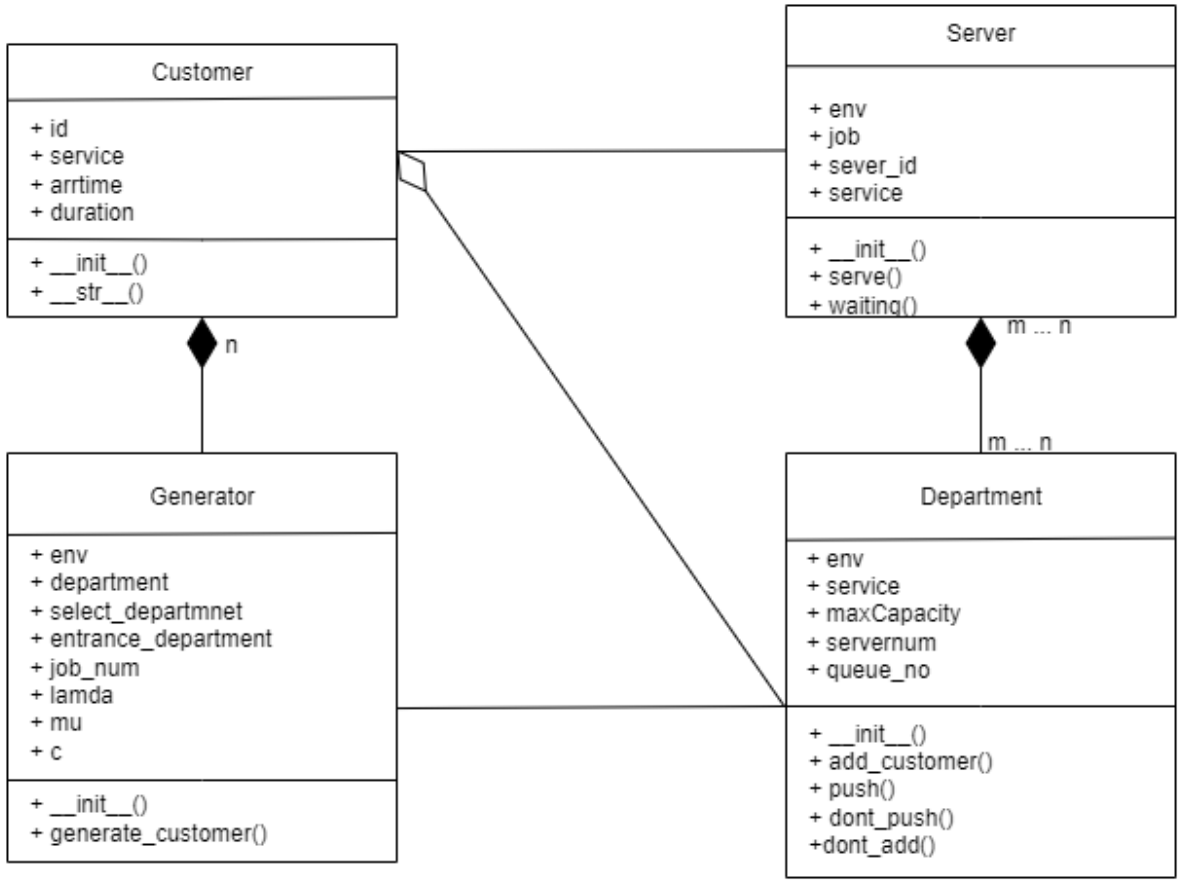

Figure 4.2: Class Diagram

4.2.3 Function Description

In **class Customer**:

- **__init__(self , id , service, arrtime, duration)**: This function is a constructor it will initialize instances of Customer class with specific attributes:
  - **self**: used to access and manipulate the instance's attributes.
  - **id**: id of customer.
  - **service**: service wants to go.
  - **arrtime**: arrive time of that customer.
  - **duration**: time to use this customer.
- **__str__(self)**: This function will print all the information of Customer has already created.

In **class Server**:

- **__init__(self , env , job , server_id , service)**: This is the constructor of this class in order to initialize instances of Customer class with specific attributes:
  - **env**: This parameter is a reference to the SimPy simulation environment where the server operates.
  - **job**: This parameter will contain instance of the class Customer.
  - **server_id**: This parameter is an identifier for the server, often used to distinguish between multiple servers in the simulation.
  - **service**: This parameter is a reference to the job or task that the server is currently serving.
- **serve(self)**:
  - This method defines the core behavior of the server. It operates in an infinity loop, simulating how the server serves jobs or tasks.
  - Within this method, the server checks whether it has a job to serve. If it does, it serves the job, calculates various metrics, and may transition the job to the next phase of the simulation.
  - If there's no job to serve, the server may go idle and enter a standby state, waiting for a job to arrive.

- **waiting(self, env)**: This method is used when the server goes idle. It sets up a timeout event to wait for a specified amount of simulation time or until interrupted.
  - **env**: The simulation environment where the server operates.

In **class Department**:

- **__init__(self, env, service, maxCapacity, servernum, queue_no)**: This function use to initialize attributes for the Department's instance. Then this function creates instances of the 'Server' class, epresenting servers within this department. The number of servers created is determined by the servernum parameter. It also starts two processes within the simulation environment that are pushing customer to server and adding customer to the department
  - **env**: This is a reference to the SimPy simulation environment where the department operates.
  - **service**: This parameter represents the type of service provided by this department. It's a distinguishing factor for the servers.
  - **maxCapacity**:This is the maximum capacity of the department. It represents the maximum number of customers the department can handle simultaneously.
  - **servernum**: The number of servers within this department.
  - **queue_no**: An identifier for the department's queue (or service queue).
- **add_customer(self)**: Method continuously checks for available space in the department's queue and adds customers from the global job list if there is space. It also handles the case where the queue is full or there are no waiting customers.
- **push(self)**: The push method continuously checks for available servers and customers in the department's queue and pushes customers to available servers, maintaining the department's customer service process. It also handles situations where there are no available servers or customers in the queue.
- **dont_push(self, env)**: It created a delay for not pushing until simulation_time end or be interrupted by the external event.
- **dont_add(self, env)**: It created a delay for not adding customer to queue until simulation_time end or be interrupted by the external event.

In **class Generator**:

- **__init__(self, env, departments, select_department, entrance_department, job_num, lamda, mu, c)**: This is the constructor of this class, which is used to assign the value to the attributes of object created.
  - **env**: This is a reference to the SimPy simulation environment where the Generator class operates.
  - **departments**: The number of department of the system, our system has 3 department which are Product guarantee team, Technical support and Sales.
  - **select_department**: The department that the customer choose to be served.
  - **entrance_department**: The department that the customer choose and now they are in the queuing node of the system and waiting to be served at random server of corresponding department.
  - **job_num**: The total customer call to be served in the system.
  - **lamda**: The arrival rate or the number of customers arriving per unit of time, each customer have different arrival time.
  - **mu**: The service rate or the number of customer being served per unit of time, service rate of each department is different.
  - **c**: the number of server for each department.
- **generate_customer(self, env)**: In this method, we have **random.expovariate()** which is the inbuilt method of random module and it is used to return the random floating point number with exponential distribution. This method help to add the customer from the waiting list to the corresponding queue node.

References

[1] Raj Jain. Wiley Computer Publishing, John Wiley & Sons, Inc. Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling, 05/01/91.

[2] Averill M. Law. Simulation Modeling and Analysis (5th edition), 2015.

[3] Micheal Baron. Probability and Statistics for Computer Scientists (2nd edition), 2014.

[4] Dennis Blumenfeld. Operation Research Calculations Handbook, 2001.