Technical Support Call Center Simulation

Grace Fujinaga, John Leigh, Timmy Li, Kevin Ou

Abstract

This project uses discrete event simulation to model a call center with the ultimate goal of understanding which types of resources are critical in providing strong customer service and decreased wait times. The model was created to model a customer seeking to set up wireless internet services. By creating this model and running sensitivity analyses it was determined that the best way to improve customer experience is by adding managers.

Problem Definition and Introduction

Customer support call centers are critical to the business operations in many different industries. Despite technical advances in software controlled queueing, automated phone systems (Interactive Voice Response and DTMF-based systems), and other technologies used to optimize the efficiency of call centers, a common complaint by end users tends to be long wait times to connect to a support agent. Our team attempted to simulate a typical call-flow by an end user into a technical support call center system by modeling the sequence of events that occur during this process using discrete event simulation methods. The model is inspired by an experience setting up wifi services. Our model's only limited resource was the number of Customer Support Representatives (CSRs) and managers that are available at any given time to handle incoming calls. Parameters include mean service time and max service time.

The graph describes the process-flow for our call center's operations by representing events in the process-flow as nodes and the transition between these events as edges connecting each node. Various trackers are in place throughout the process, for both time and queue size. The full event graph and description are in Appendix A.

Literature Review

According to Mathew and Nambiar, the call center consists of various components: the customer, PABX, IVR, ACD, SBQ and the call team. The process is initiated by the customer, then moves on to the PABX which is described as a resource that tracks the number of live calls at the moment. It is used also to make sure in case all lines are busy, calls will be blocked until a free line opens. Next is IVR, interactive voice response, where the customer interacts with a computer through the use of voice or keyboard entries. During this period the time that a customer spends before speaking to a representative is tracked. Customer service interaction may end at IVR but if not moves onto ACD, automatic call distributor. ACD is designed to route customers to representatives, if no agent is available then the call is placed in queue. In multiskill call centers, before being routed to a customer, during IVR, a customer is assigned a type which refers to the skill level that the call representative must be to handle the service. This adds another level of complexity to the call center process where a customer might need to wait for a specific level of representative.

In 1978, AT&T started developing a call processing simulator to design and evaluate call centers. At the time of Brigandi et al.'s paper, *AT&T's Call Processing Simulator (CAPS) Operational Design for Inbound Call Center,* the simulator was using discrete event simulation with both animation and queuing models. In 1992, around 2000 cases were modeled for their customers through the simulator. The overall process consisted of three steps: data collection and analysis, data case simulation, and alternative scenario simulation. The CAPS model simulates call centers in four parts, call generation, network, automatic call distributor, and call service. Simulation is used as the primary method for this process as it can be used to "model the complicated concepts of call abandonment and retrial and use distributions based upon empirical data" (Brigandi et al. 1994)

Research Design, Algorithms, Modeling Methods

This problem can be modeled with a discrete event simulation with multi-server queues. The model is in Appendix A. There is not a component of shared resources, and we did not use an agent-based simulation so every customer, service provider (sp), and manager are uniform within each class. The SimPy python library was used for the simulation to serve as the simulation clock. There is an exponential distribution used to determine the inter-arrival time. There are a variety of different parameters that can be adjusted and they are listed in Appendix B. All parameters can be tweaked. Our model was run 5 different times varying the number of service providers, managers, and varying the maximum wait time. Additionally, service time was determined using a uniform distribution for some runs and a normal distribution for other runs. These results will be discussed later. The initial parameters were set with the idea of a small technical call center support center. Personal experience was used to estimate the minimum and maximum times for the call center.

In the simulation, there are two separate queues. The customer enters the initial queue when they call the helpline. This queue is the initial help. After they are helped by an available service provider, there is a 15% chance that the call is escalated. In a real world scenario this would occur when a service provider cannot help and must pass the phone call to a manager or higher up. The second queue follows a similar structure as the first, but the service times are longer and there are fewer managers. This model also includes reneging to model if a customer hangs up after entering the queue. In this model, the customer can only renege right after joining the queue. This might be closer to balking because they renege right after joining the queue but throughout this assignment, it will be referred to as reneging. To learn which parameters most impact customer wait time, the simulation was run multiple times with different parameters to perform a sensitivity analysis.

Implementation

See the github for implementation details:

https://github.com/gracefujinaga/msds460_final_project.git

Results

See Appendix C for results from the separate runs. The figures showing the distribution of wait times, service times, and renege rate are in the github in the results folders. Additionally, an example of a set of figures is in Appendix D.

Conclusion and Management Recommendations

This project demonstrated the effectiveness of discrete event simulation in analyzing and optimizing real-world processes, specifically within the context of a call center. By modeling the flow of customer service calls through two tiers –standard service providers (SPs) and escalation managers– we were able to evaluate key performance metrics such as wait times, service times, reneging rates, and escalations under both normal and uniform service time distributions. Escalation bottlenecks, caused by limited manager availability, were a persistent challenge across all scenarios, emphasizing the need for improved resource allocation in handling escalated calls. Our findings suggest actionable recommendations for optimizing call center operations, including adopting predictable workflows using normal distribution models for service times, increasing the number of escalation managers, and leveraging predictive modeling for real-time resource allocation. These strategies can reduce inefficiencies, improve customer satisfaction, and ensure more reliable service delivery. This project also underscored the value of simulation as a decision-support tool. Discrete event simulation allows organizations to test different scenarios, identify bottlenecks, and make data-driven improvements to their processes. The insights gained through this study can be extended to other service-oriented industries, showcasing the broad applicability of simulation methods in solving complex operational challenges.

References

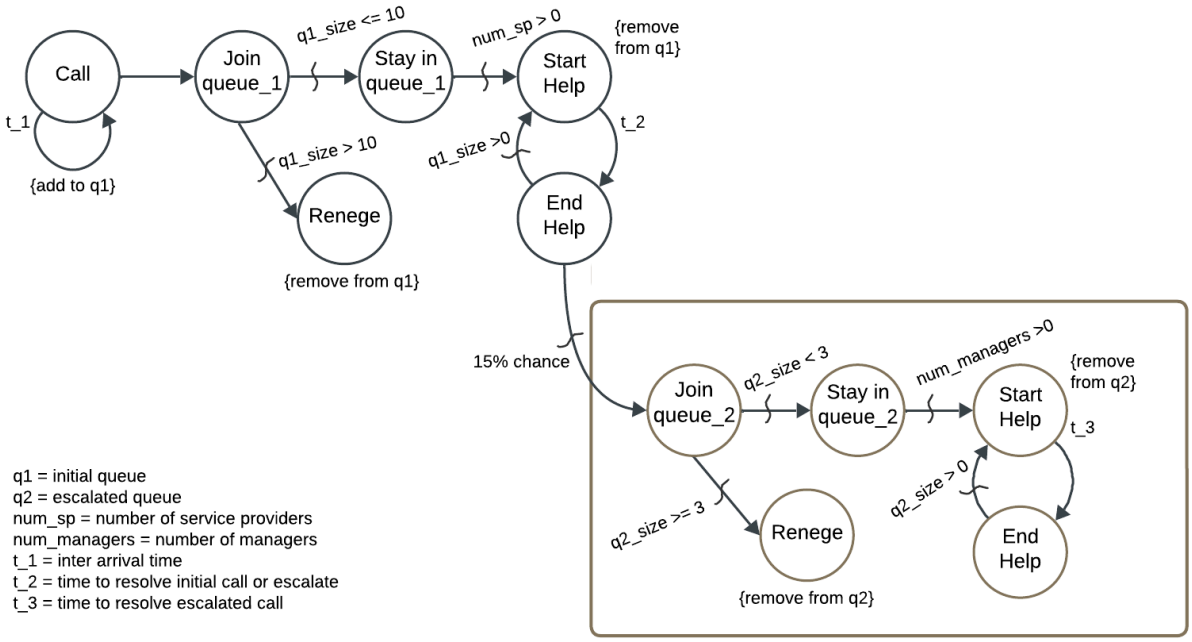
Benny Mathew, Manoj K. Nambiar (2013). A Tutorial On Modelling Call Centres Using Discrete Event Simulation, ECMS 2013 Proceedings edited by: W. Rekdalsbakken, R. T. Bye, H. Zhang, European Council for Modeling and Simulation. doi:10.7148/2013-0315

Anthony J. Brigandi, Dennis R. Dargon, Michael J. Sheehan, Thomas Spencer, III, (1994) AT&T's Call Processing Simulator (CAPS) Operational Design for Inbound Call Centers. Interfaces 24(1):6-28. https://doi.org/10.1287/inte.24.1.6

Miller, Thomas W. 2015. Modeling Techniques in Predictive Analytics with Python and R: A Guide to Data Science. Upper Saddle River, NJ: Pearson Education. [ISBN-13: 978-0-13-389206-2] Chapter 6: Operations Management, pages 81–101.

Hillier, Frederick S., and Gerald S. Liebermann. 2021. Introduction to Operations Research (eleventh ed.). New York: McGraw-Hill. [ISBN-13: 978-125987299-0] Chapter 17. Queueing Theory, pages 701–731.

Appendix A



1. A caller calls into a customer support line and joins queue 1. The initiation of the call starts t_1, inter arrival time, is logged starting here. Customers either:
    a. Renege. When queue 1 is larger than 10, the customer will renege
    b. Stay in queue 1. When queue 1 is less than 10 , the customer will stay in queue.
2. If the caller stays in queue 1, they must wait until there is an available service provider to start their call. Once the call starts between customer and service provider, the time to resolve time, t_2, is logged.
3. Once the call ends, this can be an end state. The edge directing end back to start is to represent that the service provider may start another call. There is also a chance that the customer needs their issue escalated to a higher level provider, the manager. They will then join queue 2 for this. Similar to the first queue process, the customer may either
    a. Renege. When queue 2 is larger than 3, the customer will renege
    b. Stay in queue 2. When queue 2 is less than 3 , the customer will stay in queue.
4. When there is an available manager, the call can start. Once the call between manager and customer starts, t_3 (time to resolve the escalated call) is logged. Once the call ends, we reach the end state. The edge directing end back to start is to represent that the manager may start another call.

Appendix B

Queue 1 parameters:

- num_sp = 10 # number of call center service providers
- min_service_time_1 = 2 # minutes
- mean_service_time_1 = 5 + 5 # minutes
- std_dev_service_time_1 = 1
- max_service_time_1 = 30 # minutes
- reneg_queue_length_1 = 10

Parameters for Queue 2:

- num_managers = 1
- min_service_time_2 = 10 # minutes - this is in ADDITION to previous service time, not total
- mean_service_time_2 = 20 + 5
- std_dev_service_time_2 = 10
- max_service_time_2 = 50
- reneg_queue_length_2 = 3

Overall parameters:

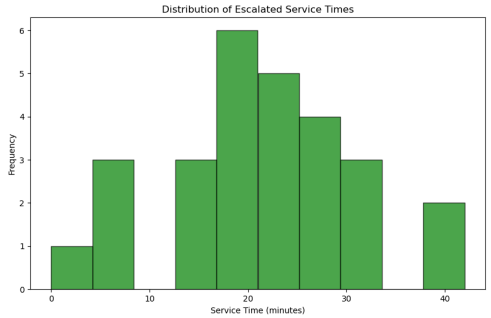
- distribution_flag = False
    - True for uniform distribution of service times, false if a normal dist for service times

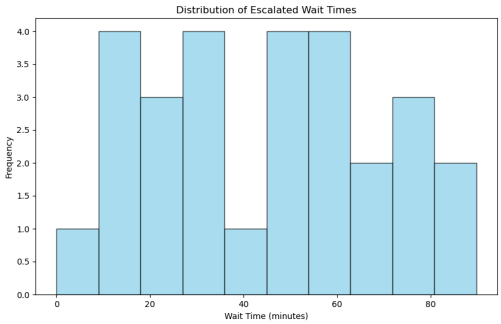
Appendix C

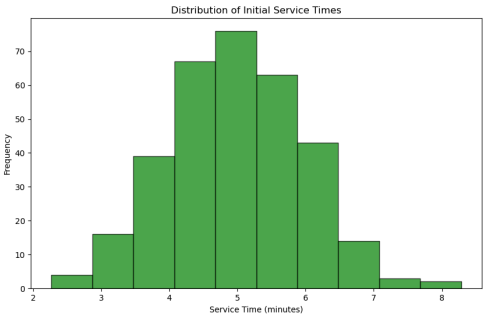
Normal Distribution (Service Time)	Total Calls	Average Wait Time (min)	Average Service Time (min)	Renege Rate (%)	Total Reneged Calls	Total Escalated calls
Run 1	329	4.26	6.70	.12	23	56
Run 2	338	1.03	7.26	.04	8	50
Run 3	319	6.88	6.66	.07	14	46
Run 4	333	3.81	6.62	.09	18	47
Run 5	328	3.60	11.56	.14	27	57

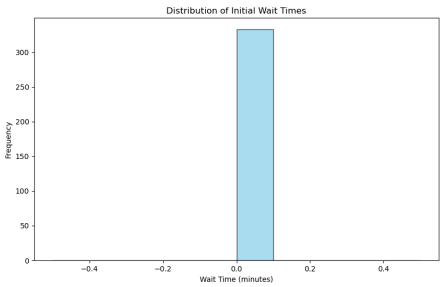
Uniform Distribution (Service Time)	Total Calls	Average Wait Time (min)	Renege Rate (%)	Total Reneged Calls	Total Escalated calls
Run 1	350	16.52	.56	118	53
Run 2	350	12.54	.32	68	45
Run 3	350	19.48	1.10	232	58
Run 4	350	4.31	.11	24	48
Run 5	350	14.22	.53	111	43

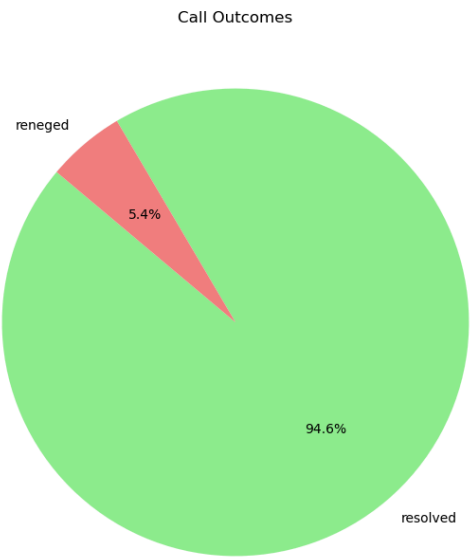
Appendix D


Distribution of Escalated Service Times
Frequency
Service Time (minutes)


Distribution of Escalated Wait Times
Frequency
Wait Time (minutes)


Distribution of Initial Service Times
Frequency
Service Time (minutes)


Distribution of Initial Wait Times
Frequency
Wait Time (minutes)


Call Outcomes
reneged
5.4%
94.6%
resolved