A Mathematical Model of Queue Management System

TM000082

1 Introduction

A mathematical model of a queue management system is needed to implement a technological solution to solve day-to-day challenges in queue handling at offices. This document serves as a foundational mathematical framework, designed to simplify the implementation of a technological solution. By breaking down complex queueing dynamics into a set of defined rules, principles, and equations, this model provides a clear blueprint for developing an effective and intelligent queue management system.

2 The Complexity of Modern Queueing Systems

At its core, an office provides certain services to its customers. The key participants are the **Service Provider**, who delivers the service, and the **Customer**, who receives it. However, the complexity of managing the interaction between them grows significantly when we consider the different types of customers that a modern system must handle:

- **Waiting Customer:** A standard walk-in customer.
- **Scheduled Customer:** A customer with a pre-booked appointment.
- **Priority Customer:** Individuals such as seniors or persons with disabilities (PWD) who are given precedence.
- **Overdue Customer:** A customer who has missed their scheduled appointment time.

The simplest queueing model involves a single service provider with a single line of walk-in customers. Complexity can be layered on by adding multiple queues for the same service. A truly complex system, however, is one that must juggle appointments, priority customers, and latecomers simultaneously.

While this document provides rules to manage these interactions, these rules can fail because real-world scenarios are not perfect. In such cases, a purely mathematical model is not the right answer; we must use computational technologies to manage the system. For instance, consider a single desk handling

multiple services with a mix of appointments, latecomers, and priority customers. This mayhem of variables creates chaos that can lead to system failure. While there should be an engineering fail-safe in the technological implementation, there cannot be a perfect mathematical fail-safe for every real-world contingency.

3 Single Queue Management

This section outlines the model for a system with a single queue and a single service provider. This is the foundational model upon which more complex scenarios will be built.

3.1 Service Time Metrics

To effectively manage the queue, it's crucial to track the time taken to serve each customer. This allows the system to build predictive metrics.

3.1.1 Average Service Time

The average time per customer is calculated as a rolling average. This provides an up-to-date measure of the service provider's performance. It is defined as the sum of all previous service durations divided by the total number of customers served.

Let T_{avg} be the average service time and T_i be the service time for the i-th customer. For n customers served, the average service time is:

$$T_{avg} = \frac{\sum_{i=1}^{n} T_i}{n} \tag{1}$$

For the very first customer ($n = 1$), a predefined base service time, T_{base}, can be used as the initial average to start the calculation.

3.1.2 Service Time Deviation

We can measure how much the service time for a particular customer deviates from the current average. This metric, which we'll denote as Δt, helps in identifying anomalies or trends in service duration. The deviation is calculated as:

$$\Delta t = T_{actual} - T_{avg} \tag{2}$$

where T_{actual} is the time taken for the current customer. This value is continuously tracked and can be used for effective resource management.

3.1.3 Root Mean Square (RMS) of Service Time

The RMS is a statistical measure that gives a higher weight to larger values. In this context, it will be more sensitive to customers who take an unusually long

time to serve. The RMS of the service times for n customers is defined as:

$$\text{RMS}_T = \sqrt{\frac{\sum_{i=1}^{n} T_i^2}{n}} \tag{3}$$

Where T_i is the service time for the i-th customer.

3.2 Queue Closure Condition

To avoid creating a queue that cannot be fully served within business hours, the system needs a rule to stop admitting new customers. The queue for walk-in customers will remain open only as long as the following condition holds true:

$$\frac{T_{rem}}{T_{avg}} > N_{queue} \tag{4}$$

Where:

- T_{rem} is the operational time remaining for the service provider.
- T_{avg} is the current average service time per customer.
- N_{queue} is the number of customers currently waiting in the queue.

This inequality ensures that the projected time to serve everyone currently in the queue is less than the time remaining in the workday. Once the number of customers in the queue is equal to or greater than the number of customers that can be served in the remaining time, no new walk-in customers are admitted.

3.3 Estimated Time to Response (ETR)

Providing an estimated wait time is a critical feature of a queue management system. The Estimated Time to Response (ETR) for the i-th customer in the queue can be calculated by considering the average service time and the actual performance of the service provider on preceding customers.

The ETR for the i-th customer is given by:

$$\text{ETR}_i = (T_{avg} \cdot (i-1)) + \sum_{j=1}^{i-1} \Delta t_j \tag{5}$$

Where:

- ETR_i is the estimated time until the i-th customer is served.
- T_{avg} is the average service time.
- i is the position of the customer in the queue.
- Δt_j is the Service Time Deviation for the j-th customer who has already been served.

This formula provides a dynamic estimate. The first term, $(T_{avg} \cdot (i-1))$, calculates a baseline ETR based purely on the average service time for the customers ahead. The second term, $\sum_{j=1}^{i-1} \Delta t_j$, adjusts this estimate by summing the total extra or less time taken for all customers who have already completed their service. This ensures the ETR adapts to the service provider's real-time performance.

3.4 System Stability and Dynamic ETR

The standard ETR calculation is effective under normal conditions. However, on days with high variability in service times (e.g., a busy Monday), the average service time, T_{avg}, may not fully capture the potential for large delays. To account for this, we can introduce a more stable, dynamic ETR that incorporates the Root Mean Square (RMS) of service times.

3.4.1 Weighted Dynamic ETR

We can now define a dynamic ETR that blends the average service time with the RMS value using a weighting factor, w. This allows the system to switch between a standard estimate and a more conservative one.

$$\text{ETR}_i^{\text{dynamic}} = [w \cdot T_{avg} + (1-w) \cdot \text{RMS}_T] \cdot (i-1) + \sum_{j=1}^{i-1} \Delta t_j \tag{6}$$

The weighting factor w is a value between 0 and 1.

- When w is close to 1, the ETR relies heavily on the average, suitable for stable, predictable days.
- When w is close to 0, the ETR gives more weight to the RMS value. This makes the estimate more conservative and accounts for greater volatility in service times.

The power of this model lies in its adaptability. By using data analytics, the system can learn to adjust the weight w based on historical data. For instance, it could automatically lower w on days that are historically busier (like Mondays) and raise it during typically quieter periods, leading to more accurate and reliable wait time predictions for customers.

3.5 Integrating Scheduled Appointments

Implementing a system for scheduled appointments is a strategic decision for the operating company. Should they choose to offer this feature, the model must be adapted to manage a hybrid queue of both walk-in customers and those with appointments.

3.5.1 Appointment Availability Condition

The system must determine if new appointments can be offered. This is based on the projected service completion time for a hypothetical new walk-in customer relative to the closing time. The number of available appointment slots, S_{avail}, can be calculated as follows:

$$S_{avail} = \left\lfloor \frac{T_{close} - (T_{now} + ETR_{N_{queue}+1})}{t_{slot}} \right\rfloor \tag{7}$$

Where:

- T_{close} is the closing time of the business (e.g., 5:00 PM).
- T_{now} is the current time.
- $ETR_{N_{queue}+1}$ is the Estimated Time to Response for a hypothetical *new* walk-in customer who would join the end of the current queue.
- t_{slot} is the duration of a single appointment slot, determined by the company.
- $\lfloor \cdot \rfloor$ is the floor function, as only full slots can be offered.

The term $(T_{now} + ETR_{N_{queue}+1})$ represents the projected time of day when the last potential walk-in would begin service. The formula calculates the remaining time in the workday after this point and determines how many appointment slots can fit. New appointments can be offered only if $S_{avail} > 0$.

Scope of Availability It is important to note that this formula is intended for booking available slots on the **present day only**. It ensures that appointments are not booked beyond the closing time of the current business day. The process for scheduling appointments for future days is considered a separate function and is not governed by this real-time availability calculation.

3.5.2 Calculating Walk-in ETR with Appointments

When the system must handle both walk-ins and appointments, a walk-in's Estimated Time of Response (ETR) must be calculated **iteratively** to be accurate. The wait time is repeatedly adjusted to account for the "chain reaction" of delays caused by scheduled appointments until the final ETR becomes stable. The core logic is to find a "stable" ETR where adding the delay from all preceding appointments no longer pushes the ETR past any new appointments.

Definitions:

$ETR_{\mathbf{base}}$ The initial ETR for the walk-in, based only on the existing walk-in queue.

$t_{\mathbf{slot}}$ The fixed duration of one appointment slot (e.g., 20 minutes).

$T_{\textbf{now}}$ The current time when the walk-in requests service.

$m_{\textbf{final}}$ The **final, stable count** of appointments between T_{now} and the final, projected service time ($T_{\text{now}} + ETR_{\text{final}}$).

Final State Formula: The process is iterative, but the final, correct ETR will aatisfy this equation:

$$ETR_{\text{final}} = ETR_{\text{base}} + m_{\text{final}} \cdot t_{\text{slot}} \tag{8}$$

Finding m_{final} requires an iterative algorithm where the ETR is calculated, the number of appointments within that new timeframe is counted, and the ETR is recalculated with the new appointment count until the number of appointments no longer increases.

3.6 Managing Overdue Customers (Latecomers)

A fair and flexible system must have a clear policy for customers who arrive after their expected service time. The management of a late customer is handled with a dynamic approach triggered by their check-in status.

3.6.1 Handling Walk-in Latecomers

The process is divided into two phases.

Phase 1: Customer is Late (Not Checked In) When a customer's turn arrives but they have not checked in, they are marked as 'Late' and temporarily skipped. This immediately shortens the ETR for all subsequent customers, as the queue is effectively one person shorter.

Phase 2: Late Customer Arrives (Checks In) When the late customer physically checks in, their new position in the queue is determined by a configurable company policy, represented by a **Lateness Placement Factor** (W_L). This factor, a value between 0 and 1, allows a business to choose how strictly to handle latecomers.

- $W_L = 1$ **(Strict Policy):** The customer is sent to the very end of the line.
- $W_L = 0$ **(Lenient Policy):** The customer is placed at the front of the waiting queue (immediately after the person currently being served).
- $0 < W_L < 1$ **(Hybrid Policy):** The customer is placed proportionally along the length of the queue. A value of $W_L = 0.5$ would place them in the middle of the current line.

The new position, P_{new}, is calculated as follows:

$$P_{new} = \text{round}(1 + W_L \cdot N_{queue}) \tag{9}$$

Where N_{queue} is the number of customers currently in the queue. Once the late customer is re-inserted at this new position, the ETR for all subsequent customers is recalculated.

Grace Period for Walk-ins To prevent penalties for minor delays, a grace period is implemented. If a customer checks in within this window, they retain their original spot. The grace period is calculated dynamically:

$$T_{grace} = (T_{avg} \cdot w_{grace}) + T_{base\_grace} \tag{10}$$

Where T_{avg} is the average service time, w_{grace} is a weighting factor (< 1) adjusted via data analytics, and T_{base_grace} is a fixed base grace time.

3.6.2 Handling Appointment Latecomers

Customers who miss their scheduled appointment are also granted a grace period, calculated with the same formula as for walk-ins. If they arrive after their grace period has expired, they are given two options to ensure fairness to punctual customers:

1. **Join the Walk-in Queue:** Forfeit their appointment and enter the queue as a new walk-in customer.
2. **Reschedule:** Book the next available future appointment slot.

3.7 Managing Priority Customers

This model handles priority customers by physically inserting them into the queue at a position determined by a pre-defined ratio, ensuring a fair and predictable wait for everyone.

3.7.1 Logic: Priority Insertion System

A company can define multiple levels of priority (e.g., for seniors, persons with disabilities, expectant mothers), each with its own configurable ratio, R. This ratio dictates the guaranteed spacing of that priority type within the queue. For example, a ratio of $R = 4$ means there will be at least 3 regular customers between each customer of that priority type.

3.7.2 Insertion Algorithm

When a priority customer arrives, their position in the queue is determined algorithmically. This ensures the correct spacing is maintained and the queue is reordered transparently.

Algorithm Steps:

1. **Identify Type and Ratio:** When a priority customer of type X with ratio R_X arrives, the system identifies their specific rules.
2. **Find Last Position:** The system scans the current queue from front to back to find the position of the last customer who is also of type X. Let this position be P_{last_X}.
3. **Calculate New Position:** The insertion position for the new customer, P_{new}, is calculated.
   - If a customer of type X was found, the new position is $P_{new} = P_{last\_X} + R_X$.
   - If no customer of type X was found in the queue, the new position is $P_{new} = R_X$.
4. **Insert and Reorder:** The new priority customer is inserted at position P_{new}. All subsequent customers are shifted back one spot. If the calculated position P_{new} is greater than the current length of the queue, the customer is simply placed at the end.

4 Multi-Queue Management

The model is extended to a system with multiple officers, each managing a separate queue but providing the identical service. This introduces the challenge of distributing customers and managing potential imbalances between the queues.

4.1 Customer Distribution

A multi-queue system requires a sophisticated method for assigning customers to a queue.

4.1.1 Initial Assignment: Round-Robin

When customers first arrive, they are assigned to the available queues in a round-robin fashion to ensure an initially balanced distribution. However, this can lead to imbalances if one officer is significantly slower than others.

4.1.2 Dynamic Assignment: Lowest ETR Principle

To overcome this limitation, a dynamic principle is introduced: **A new customer is always assigned to the queue with the lowest ETR for them.** The system calculates their potential ETR for each queue and directs them to the one providing the shortest estimated wait time. This ensures customers are routed to the most efficient service point at that moment.

4.2 Managing Different Customer Types

The principles for handling scheduled, priority, and overdue customers can be extended to the multi-queue environment through effective technological implementation. For example, a business can leverage technology to dedicate specific queues for specific functions (e.g., one for appointments, one for walk-ins) or apply priority logic globally. The core ETR and service time metrics for each individual queue remain the same. The key is the intelligent, system-level application of these rules.

4.3 Intelligent Reshuffling for Queue Balancing

Even with intelligent initial assignment, variations in service times can cause queues to become imbalanced. To counteract this, the system employs a continuous optimization algorithm to dynamically rebalance the queues. The goal is not simply to equalize queue lengths, but to minimize the **Total System ETR** (the sum of ETRs for all customers).

Algorithm Logic

1. Calculate the current Total System ETR.
2. For each customer in the system, simulate moving them to every other available empty slot (in their current queue or another queue).
3. For each simulated move, recalculate what the new Total System ETR would be.
4. If a move is found that results in a lower Total System ETR, make that move permanent.
5. Repeat this process until no single customer move can be found that further reduces the Total System ETR. The system is now in an optimized state.

This "smart shuffling" method ensures that customers are arranged in a way that is demonstrably more efficient for the group as a whole, providing a gentle and continuous rebalancing mechanism.

5 Conclusion

This document has detailed a comprehensive mathematical model for a modern queue management system, progressing from a foundational single queue to a complex, multi-queue environment. The model incorporates a dynamic ETR, configurable policies for diverse customer types (including appointments, priority, and latecomers), and intelligent algorithms for customer assignment and queue balancing. The principles and equations outlined provide a robust and

adaptable framework that balances operational efficiency with customer fairness, serving as a solid foundation upon which a powerful and practical queue management technology can be built.