Optimal Order Quantity for Football Jersey Inventory Under Normal and Seasonal Demand Using FlexSim

Anan Butrat[1] and Chutikarn Suppatvech[1*]

[1]Department of Management Engineering, Faculty of Industrial Technology, Valaya Alongkorn Rajabhat University under the Royal Patronage

Corresponding Author Email : chutikarn.su@vru.ac.th

Received 25 March 2025

Revised 15 June 2025

Accepted 14 August 2025

Abstract

Effective inventory management is critical for small to medium-sized (SMEs) retail businesses as they are usually facing demand fluctuations, especially for seasonal or promotional items. A case study of a football jersey retail shop in Pathum Thani has also experienced such a challenge, where they have relatively stable demand in normal months and fluctuating demand during football competition months. The shop typically orders a fixed quantity of 300 units during the normal period and 600 units during the seasonal period. However, this order strategy frequently leads to stockouts, resulting in lost sales opportunities and dissatisfied customers. Accordingly, the purpose of this study is to optimize and propose the order quantity strategy that balances stock levels. This study employs FlexSim, a discrete-event simulation software, to analyze and evaluate 14 scenarios of different order quantity strategies, testing different possible combinations of normal and seasonal order quantity values. The results show that Scenario 14 with a normal order quantity of 550 units and a seasonal order quantity of 950 units provided the optimal order quantity with 99.87% Releasing, 0.13% Empty, and no remaining inventory. Based on the results, this order quantity strategy minimizes stockouts while preventing excess stock accumulation, making it the most optimized order policy among the other 13 scenarios. The findings also highlight the importance of dynamic inventory control in optimizing inventory replenishment, where discrete-event simulation software (i.e., FlexSim) can be utilized by the SMEs in the retail industry in fine-tuning order quantity values corresponding to demand fluctuations. Hence, retail businesses can improve customer satisfaction through enhancing inventory efficiency.

Keywords: Reorder Point, Inventory Optimization, Seasonal Demand, FlexSim, Football Jersey Sales

1. Introduction

Effective inventory management is essential for balancing product availability with operational efficiency, particularly in retail environments where products (e.g., clothing, food and beverages and household goods) are subject to variable customer demand. The key challenge for retailers is to respond to such demand fluctuations, especially for seasonal or event-driven demand spikes that can lead to either excess inventory or frequent stockouts. This is primarily critical for small to medium-sized enterprises (SMEs), which may lack advanced forecasting systems and rely on fixed ordering strategies that are often unresponsive to rapid demand surges [1]–[2].

A case study example in this research focuses on a small-sized clothing retailer, the football jersey shop in Pathum Thani, Thailand, which experiences a predictable yet significant increase in customer demand during local football competition months. The historical sales of football jersey data are collected for 90 days, covering both periods of normal and seasonal demand as shown in Table 1. According to the previous data, under regular conditions, the shop sells approximately 10 pieces per day. Hence, the shop places a monthly order of 300 units, which suffices for normal demand. During competition months, the shop sells approximately double the amount per day. Hence, the order quantity has increased to 600 units to ensure stock availability. Despite this proactive measure, the store continues to experience stockouts, indicating a mismatch between static inventory policies and dynamic, time-dependent demand fluctuations. These stockouts result in lost sales and reduced customer satisfaction.

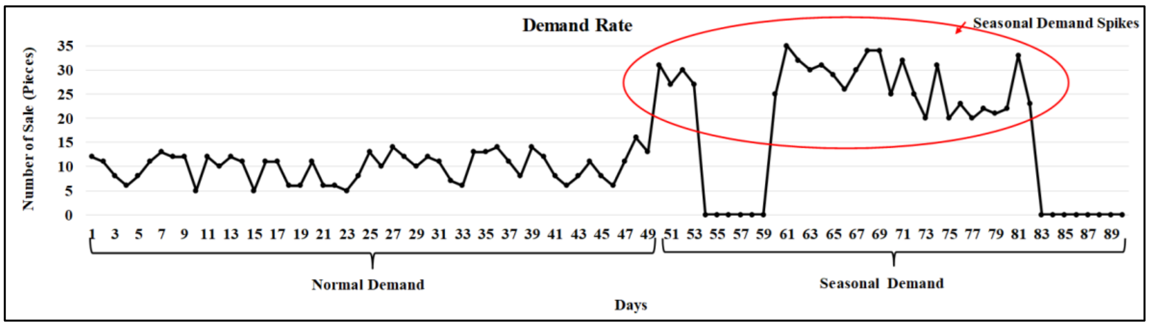


Fig. 1 Demand rate of a football jersey shop in Pathum Thani, Thailand

Although previous research has addressed inventory control under uncertain or fluctuating demand—often using techniques such as Economic Order Quantity (EOQ), safety stock models, stochastic demand modeling, and simulation-based optimization [3]–[6]—many of these studies focus on large-scale or manufacturing systems. Several works also apply simulation to evaluate inventory performance under probabilistic demand [7]–[8], yet few have incorporated simulation control mechanisms, such as time-based triggers, to precisely replicate real-world demand shifts within simulation environments.

To address this gap, this research introduces a novel approach by using FlexSim's discrete-event simulation (DES) capabilities in conjunction with time-based port control to simulate seasonal demand patterns accurately. DES allows modeling of a system as a chronological sequence of discrete events—each event triggering a change in system state [9]. Specifically, this model employs time-triggered logic to switch between normal and seasonal demand flows.

This study aims to evaluate and optimize an order quantity strategy that minimizes stockouts while reducing excess inventory, ultimately enhancing service levels in a seasonal retail context. This research contributes to the field by demonstrating a flexible simulation methodology that can be adapted for other small businesses facing cyclical or event-driven demand fluctuations [10]–[11]. Moreover, the study proposes a simulation control algorithm that dynamically switches between normal and seasonal demand modes using time-based and event-triggered control within FlexSim.

2. Literature Review

This study proposes an algorithm or logic for switching demand that incorporates both event-triggered control strategies and time-triggered logic, as below.

Event-triggered control strategies have been developed to optimize system performance by initiating control actions based on specific events rather than at fixed time intervals. This approach reduces unnecessary computations and communications, thereby conserving resources. For instance, Zhang et al. (2022) [12] proposed an event-triggered control algorithm for discrete-time delay systems, where control inputs are updated only when a measurement error surpasses a dynamic threshold, ensuring system stability while minimizing control updates. Similarly, Xu et al. [13] introduced a state-segmentation-based event-triggered mechanism for stabilizing discrete-time 2-D switched systems. Their method segments the state space and applies control actions only when the system state enters specific regions, effectively balancing control performance and resource utilization.

Time-triggered logic involves initiating actions at predetermined time instances, which is crucial for systems requiring synchronization and adherence to temporal specifications. Hashimoto and Dimarogonas [14] explored resource-aware networked control systems under temporal logic specifications, emphasizing the importance of time-triggered strategies in ensuring system behaviors align with temporal constraints. Moreover, Maity and Baras [15] proposed an event-triggered control framework for dynamical systems with temporal logic constraints, ensuring that system trajectories satisfy specified temporal properties while reducing communication loads.

In contrast, this research introduces a novel approach that combines event-triggered and time-triggered logic within a simulation context to control inventory flow. Specifically, it proposes a time-based simulation control algorithm that dynamically switches between normal and seasonal demand modes in FlexSim by activating or deactivating ports according to simulation time. This fills a critical gap in the literature by demonstrating how simulation time can be leveraged not only for event scheduling but also for operational decision-making in demand-driven inventory systems. The summary of the literature review and comparison is shown in Table 1.

Table 1 Literature review summary

Study	Control Strategy	Simulation Context	Time-Based Port Control	Simulation Time Integration	Application Focus
Zhang et al. [12]	Event-triggered control	Discrete-time control system			Delay system stabilization
Xu et al. [13]	Event-triggered control	2-D switched systems			System stabilization
Hashimoto and Dimarogonas [14]	Time-triggered control	Networked control systems		✓	Temporal logic constraints
Maity and Baras [15]	Event-triggered control	Dynamical systems		✓	Communication reduction
This Research	Time-based and Event-triggered control	FlexSim (DES platform)	✓	✓	Retail inventory control with demand switching

While this research employs FlexSim as the core simulation platform due to its discrete-event, object-oriented architecture and built-in port control capabilities, it is important to compare it with other widely used simulation tools to highlight its advantages as shown in Table 2. AnyLogic, developed by The AnyLogic Company (2023), supports hybrid simulation (discrete-event, agent-based, and system dynamics) and allows flexible control logic using Java programming. Although powerful, its reliance on code for time/event logic increases the development burden for non-programmers. Arena, developed by Rockwell Automation (2023), and Simul8, developed by Simul8 Corporation (2023), are popular for process-based discrete-event simulation. Both are user-friendly and suitable for general process flow analysis; however, their native support for dynamic, time-based port control is limited and typically requires custom logic or indirect configuration. In contrast, FlexSim (FlexSim Software Products, Inc., 2023) provides direct visual and logic-based control of ports based on simulation time or custom triggers, making it highly effective for modeling demand-switching inventory systems where real-time flow control is essential.

Table 2 Comparison of simulation software for time-based and event-triggered control

Feature / Tool	FlexSim [16]	AnyLogic [17]	Arena [18]	Simul8 [19]
Modeling Approach	Discrete-event, object-oriented	Hybrid (DES, ABS, SD)	Discrete-event	Discrete-event
Time-based Control	Native support (via triggers)	Supported via Java scripting	Limited (requires customization)	Limited (via process configuration)

Feature / Tool	FlexSim [16]	AnyLogic [17]	Arena [18]	Simul8 [19]
Event-triggered Control	Built-in with triggers & logic	Programmable via Java	Not natively supported	Limited
Port Control Logic	Intuitive, graphical + logic	Scripted control possible	Not natively available	Requires workaround
Ease of Use	Moderate (requires some logic)	Complex (requires coding)	High (drag-and-drop)	High (drag-and-drop)
Best Use Case	Dynamic object control, inventory	Complex systems with hybrid logic	Process flow analysis	Simple process modeling

3. Methodology

3.1 Data Collection

This study collects football jersey sales data for 90 days to analyze demand variations between normal and seasonal conditions from a shop in Pathum Thani, Thailand. The first 49 days (excluding 6 days that were out of stock) represent normal demand, where sales occur at a regular rate, while the 27 days (excluding 8 days that were out of stock) correspond to a seasonal demand spike due to a football competition. The interarrival time is calculated by dividing the total minutes in a day by the total jerseys sold that day (Equation 1), providing insights into daily sales frequency as demand rate per day. The summary of demand and interarrival time on each day is shown in Table 3.

$$\text{Interarrival Time} = \frac{24\, hrs\ \times 60\, mins}{Total\ daily\ sale} \tag{1}$$

Table 3 Demand and interarrival time on each day

Normal		Seasonal	
Demand (Pieces)	Interarrival Time (Minutes per Piece)	Demand (Pieces)	Interarrival Time (Minutes per Piece)
12	120.0000	**31**	46.4516
11	130.9091	**27**	53.3333
8	180.0000	**30**	48.0000
6	240.0000	**27**	53.3333
8	180.0000	**25**	57.6000
11	130.9091	**35**	41.1429
13	110.7692	**32**	45.0000
12	120.0000	**30**	48.0000
12	120.0000	**31**	46.4516

Normal		Seasonal	
Demand (Pieces)	Interarrival Time (Minutes per Piece)	Demand (Pieces)	Interarrival Time (Minutes per Piece)
5	288.0000	**29**	49.6552
12	120.0000	**26**	55.3846
10	144.0000	**30**	48.0000
12	120.0000	**34**	42.3529
11	130.9091	**34**	42.3529
5	288.0000	**25**	57.6000
11	130.9091	**32**	45.0000
11	130.9091	**25**	57.6000
6	240.0000	**20**	72.0000
6	240.0000	**31**	46.4516
11	130.9091	**20**	72.0000
6	240.0000	**23**	62.6087
6	240.0000	**20**	72.0000
5	288.0000	**22**	65.4545
8	180.0000	**21**	68.5714
13	110.7692	**22**	65.4545
10	144.0000	**33**	43.6364
14	102.8571	**23**	62.6087
12	120.0000		
10	144.0000		
12	120.0000		
11	130.9091		
7	205.7143		
6	240.0000		
13	110.7692		
13	110.7692		
14	102.8571		
11	130.9091		
8	180.0000		
14	102.8571		
12	120.0000		

Normal		Seasonal	
Demand (Pieces)	Interarrival Time (Minutes per Piece)	Demand (Pieces)	Interarrival Time (Minutes per Piece)
8	180.0000		
6	240.0000		
8	180.0000		
11	130.9091		
8	180.0000		
6	240.0000		
11	130.9091		
16	90.0000		
13	110.7692		

3.2 Data Fitting and Distribution Selection

To accurately model demand patterns in FlexSim, interarrival time data (Table 1) were analyzed using a program named “Input Analyzer”, which identified the best-fitting probability distributions for interarrival times under both normal and seasonal demand conditions. For normal demand, the Weibull distribution was the best fit, expressed as 90 + WEIB (74.5, 1.15) as demonstrated in Figure 2. For seasonal demand, the Beta distribution was the best fit, expressed as 41 + 31 × BETA (0.571, 0.753) as demonstrated in Figure 3.

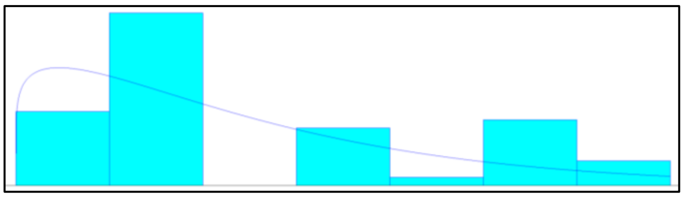


Fig. 2 Data fitting of normal demand

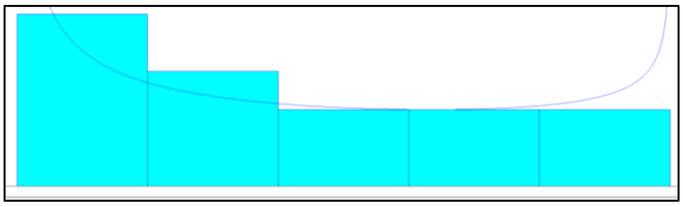


Fig. 3 Data fitting of seasonal demand

3.3 Simulation Model

FlexSim is a powerful discrete-event simulation (DES) software designed to model, analyze, and optimize real-world systems in industries such as manufacturing, logistics, healthcare, and retail. It allows users to build virtual models of processes using drag-and-drop 3D objects and customize behavior through logic and scripting. FlexSim simulates the flow of items or entities through a system based on event sequences and time progression, enabling detailed performance analysis under various conditions. With support for statistical distributions, random number generation, and real-time animation, FlexSim helps identify bottlenecks, test operational changes, and improve decision-making without disrupting actual operations.

FlexSim generates random numbers using built-in statistical distribution functions to simulate real-world variability in system behavior. These functions, such as uniform(a, b), normal(mean, stddev), exponential(mean), and others, are used to model processing times, arrival rates, and other stochastic elements in a simulation. The random number generator in FlexSim relies on a pseudo-random number algorithm seeded by the system clock or a user-defined value to ensure reproducibility. Users can control randomness through the use of distributions in process flow objects or through custom logic using the FlexScript language, allowing for accurate modeling of uncertainty and variability in discrete-event simulations.

FlexSim was used to help with model design by applying discrete-event simulation (DES) to replicate the real-world operations of the football jersey shop. Discrete-event simulation models a system as a sequence of distinct events that occur over time, such as customer arrivals, stock level changes, and reorder events. In the FlexSim model, each object—such as the source, queue, processors (Normal demand and Seasonal demand), and sink—represents a key element of the shop's workflow as shown in Figure 4. Customer arrivals, driven by probability distributions, trigger events that flow through the system, activating demand processors based on the current simulation time. FlexSim handles each event chronologically, updating inventory levels and controlling port behaviour via time-based triggers.

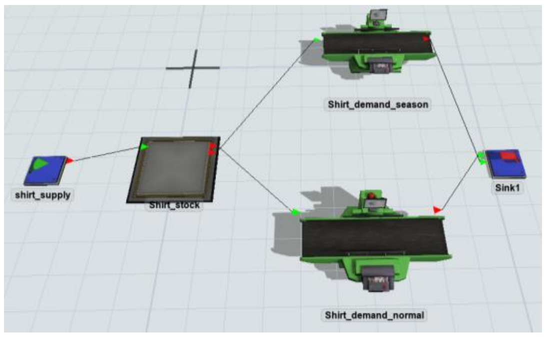
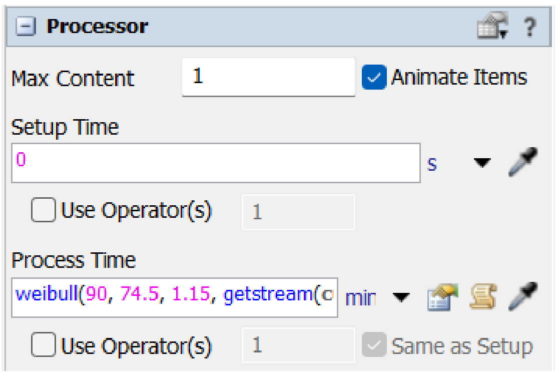


Fig. 4 Optimal order quantity model using FlexSim

The processors (Normal demand and Seasonal demand) are the first two objects for setting. After fitting the demand distributions, the FlexSim simulation model was designed to replicate the operational workflow of the football jersey shop, integrating customer arrivals, inventory tracking, and stock replenishment strategies. Normal demand followed a Weibull distribution (90 + WEIB (74.5, 1.15)) as shown in Figure 5, while seasonal demand used a Beta distribution (41 + 31 × BETA (0.571, 0.753)), ensuring realistic demand variations.



Processor

Max Content 1 ☑ Animate Items

Setup Time
0 s

☐ Use Operator(s) 1

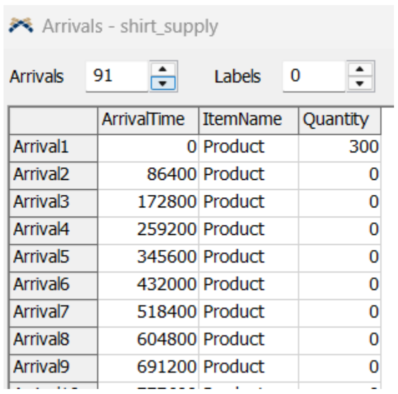
Process Time
weibull(90, 74.5, 1.15, getstream(c min

☐ Use Operator(s) 1 ☑ Same as Setup

Figure 5 Process time of the normal demand port

Moreover, the source is the second object because it is the key for setting the order quantity. The arrival time was set as one per day, and the quantity on the days that are not reorder days was set to zero as shown in Figure 6. In addition, for the days that are reorder days, the normal (pre-seasonal) period refers to the period before the start of the seasonal demand. In this study, the order quantities are replenished three times, representing the first day of each of three consecutive months. When aligning this with the simulation timeline, the start of normal demand corresponds to day 0 (0 seconds of simulation time), the pre-seasonal day occurs on day 31 (2,678,400 seconds), and the start of the seasonal demand falls on day 60 (5,184,000 seconds) of simulation time.

Furthermore, an object named "queue" is used to receive entities from the source, while the sink is responsible for removing the entities from the system. No additional settings are applied to either object.



Arrivals - shirt_supply

Arrivals 91 Labels 0

	ArrivalTime	ItemName	Quantity
Arrival1	0	Product	300
Arrival2	86400	Product	0
Arrival3	172800	Product	0
Arrival4	259200	Product	0
Arrival5	345600	Product	0
Arrival6	432000	Product	0
Arrival7	518400	Product	0
Arrival8	604800	Product	0
Arrival9	691200	Product	0

Fig. 6 Process time of the normal demand port

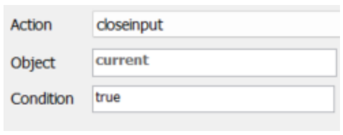
3.4 Time-based and Event-triggered control

In this study, time-based and event-triggered control is proposed for switching Processors between normal and seasonal demand. The starting point of the seasonal period is set at day 49 or 4,233,600 seconds from the start, and the end point of the seasonal period is at day 88 or 7,603,200 seconds from the start. The triggers were set on both the Normal Demand Port (Figure 8) and Seasonal Demand Port (Figure 9) to activate or deactivate them based on the simulation time. In addition, the time-based and event-triggered control is set as a loop, including the normal demand from day 0 to day 49, then changing to seasonal demand from day 50 to day 88, and then changing back to normal demand from day 89 to day 90.

For normal demand, an On Entry trigger was configured to close the input to the seasonal demand port when Time ≥ 7,603,200 seconds, ensuring that arrivals remained within the normal demand period as Figure 7(d). Conversely, an On Exit trigger opened the seasonal demand port when 4,233,600 seconds ≤ Time ≤ 7,603,200 seconds, allowing the transition to seasonal demand during the competition period as Figure 7(e).

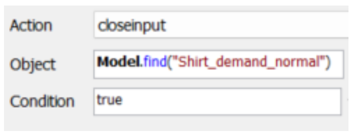
Similarly, the Seasonal Demand Port was equipped with an On Entry trigger to close the input to the normal demand port once activated, preventing overlap between demand types as Figure 7(b). When the seasonal period ended, an On Exit trigger reopened the normal demand port when Time ≥ 4,233,600 seconds, switching back to the regular demand pattern as Figure 7(c).

Additionally, at the start of the simulation, an On Simulation Start trigger was implemented to close the input to the currently active port, ensuring that demand conditions were controlled from the beginning as Figure 7(a).



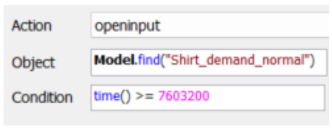
Action	closeinput
Object	current
Condition	true

(a) On simulation start of Seasonal



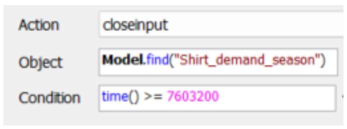
Action	closeinput
Object	Model.find("Shirt_demand_normal")
Condition	true

(b) On entry of Seasonal



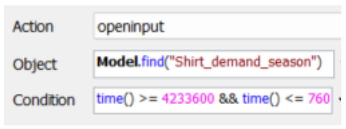
Action	openinput
Object	Model.find("Shirt_demand_normal")
Condition	time() >= 7603200

(c) On exit of Seasonal



Action	closeinput
Object	Model.find("Shirt_demand_season")
Condition	time() >= 7603200

(d) On entry of Normal



Action	openinput
Object	Model.find("Shirt_demand_season")
Condition	time() >= 4233600 && time() <= 760

(e) On exit of Normal

Fig. 7 Trigger control of normal and seasonal demand processor

```
Algorithm of Normal demand processor
Simulation time = 0
If Simulation Time >= Start point of the Seasonal demand and Simulation Time <= End point of the Seasonal demand:
    Seasonal demand processor is opened
If Simulation Time >= End point of the Seasonal demand:
    Seasonal demand processor is closed
Simulation time = Stop time
```

Fig. 8 Algorithm of normal demand processor

```
Algorithm of Seasonal demand processor
Simulation time = 0
Seasonal demand processor is closed
While Entity is Entry in Seasonal demand processor:
    Normal demand processor is closed
If Simulation Time >= End point of the Seasonal demand:
    Normal demand processor is opened
Simulation time = Stop time
```

Fig. 9 Algorithm of seasonal demand processor

3.5 Model Validation

To validate the simulation model, a paired t-test was conducted to compare real and simulated stock values across 13 weeks (7 days for 12 weeks and 6 days for last week, in total 90 days). The assumption is to compare the values of stock each week between the demand rate of the real and the demand rate from the model that represented the best-fitting probability distributions for interarrival times under both normal and seasonal demand conditions and the proposed algorithms of time-based and event-triggered control. The null hypothesis assumes no significant difference between the means, while the alternative hypothesis suggests a difference. With a significance level (α) of 0.05, the calculated t-value (-0.7944) falls within the acceptance region (-2.179 to 2.179), leading to the acceptance of the null hypothesis as shown in Table 4. This indicates that the simulation model effectively represents the real stock values with no statistically significant difference.

Table 4 Comparing the values of stock each week

No.	Real	Sim	Diff	Hypothesis Testing
1	231	237	-6	$\bar{x}_d = -6.2308$
2	157	171	-14	$SD_d = 28.2788$
3	101	112	-11	$n_d = 13$
4	33	47	-14	
5	261	284	-23	$H_0: \mu - \bar{x}_d = 0$
6	188	221	-33	$H_1: \mu - \bar{x}_d \neq 0$
7	115	159	-44	$\mathrm{df} = 12$
8	0	0	0	$\alpha = 0.05$
9	478	496	-18	p-value = 2.179
10	269	262	7	t = -0.7944
11	98	23	75	
12	0	0	0	$\therefore$ -2.179 $\geq$ -0.7944 $\leq$ 2.179

13	0	0	0	(Accept H_0)

3.6 Simulation Scenarios

The current scenario (Scenario C) represents the shop's current inventory policy, where 300 units are ordered in normal months and 600 units during seasonal months. To improve inventory performance and reduce stockouts, additional scenarios are simulated by incrementally increasing the order quantity during the seasonal period while keeping the normal monthly order fixed at 300 units. However, once the seasonal order quantity reaches its limit (1000 units), further inventory adjustments focus on increasing the normal (pre-seasonal) order quantity to compensate for potential stock shortages and ensure continuous availability.

The optimal scenario is determined by identifying the balance point between supply and demand that minimizes both lost sales and excess inventory. Effective decision-making is guided by evaluating simulation outcomes, where the best scenario is characterized by the highest number of units successfully released to customers (Releasing), along with the lowest values of empty queue instances (Empty) and remaining last stock (Last stock). These indicators collectively reflect an efficient inventory management strategy that meets demand without overstocking.

4. Results

4.1 Optimal Order Quantity Strategy

The simulation results provide a comparative analysis of different order quantity strategies under normal and seasonal demand conditions. Table 5 summarizes the outcomes across multiple scenarios, focusing on key performance metrics: Releasing (percentage of demand fulfilled), Empty (percentage of stockout occurrences), and Last Stock (remaining inventory at the time of reorder).

Table 5 Inventory status and last stock of each scenario

Scenarios	Order Quantity		Inventory Status		Last stock
	Normal (pre-seasonal)	Seasonal	Releasing	Empty	
C	300	600	79.39	20.61	0
1	300	650	81.07	18.93	0
2	300	700	82.77	17.23	0
3	300	750	84.48	15.52	0
4	300	800	86.02	13.98	0
5	300	850	87.64	12.36	0
6	300	900	89.24	10.76	0
7	300	950	90.84	9.16	0
8	300	1000	93.04	6.96	32

Scenarios	Order Quantity		Inventory Status		Last stock
	Normal (pre-seasonal)	Seasonal	Releasing	Empty	
9	350	1000	94.58	5.42	36
10	400	1000	96.21	3.79	36
11	450	1000	97.8	2.2	37
12	500	1000	99.38	0.62	37
13	550	1000	100	0	72
14	550	950	99.87	0.13	0

In the baseline scenario (C), where the order quantity was set at 300 units for normal demand and 600 units for seasonal demand, the Releasing rate was 79.39%, while Empty occurrences were at 20.61%, indicating significant stockout issues. As the seasonal order quantity was increased incrementally from 650 to 1000 units (Scenarios 1–8), the Releasing percentage steadily improved, and Empty occurrences decreased. For instance, at 850 units (Scenario 5), Releasing increased to 87.64%, and Empty reduced to 12.36%. When the seasonal order quantity reached 1000 units (Scenario 8), the Releasing rate peaked at 93.04%, but a Last Stock of 32 units was observed, indicating potential overstocking.

To further optimize inventory performance, the normal order quantity was increased beyond 300 units while maintaining a seasonal order quantity of 1000 units (Scenarios 9–13). Increasing the normal order quantity to 350 units (Scenario 9) led to a Releasing rate of 94.58%, while Empty occurrences dropped to 5.42%. Further increasing the normal order quantity to 450 and 500 units (Scenarios 11–12) resulted in significant improvements, with Releasing reaching 99.38%, while Empty occurrences were minimized to 0.62%.

The best-performing scenario (Scenario 13), where the normal order quantity was 550 units and the seasonal order quantity was 1000 units, achieved 100% Releasing with 0% Empty occurrences. However, a Last Stock of 72 units was recorded, indicating excess inventory. To balance inventory efficiency while maintaining high fulfillment rates, Scenario 14 (normal order quantity: 550, seasonal order quantity: 950) provided an alternative with 99.87% Releasing, only 0.13% Empty, and no remaining inventory. This scenario minimized stockouts while preventing excess stock accumulation, making it the most optimized inventory policy.

4.2 Comparison to Traditional Queueing System

The comparison between the Traditional Queueing System simulation and the simulation using Time-Based and Event-Triggered Control reveals significant differences in accuracy when predicting real values based on the assumption from the model validation that compares the values of stock each week between the demand rate of the real and the demand rate.

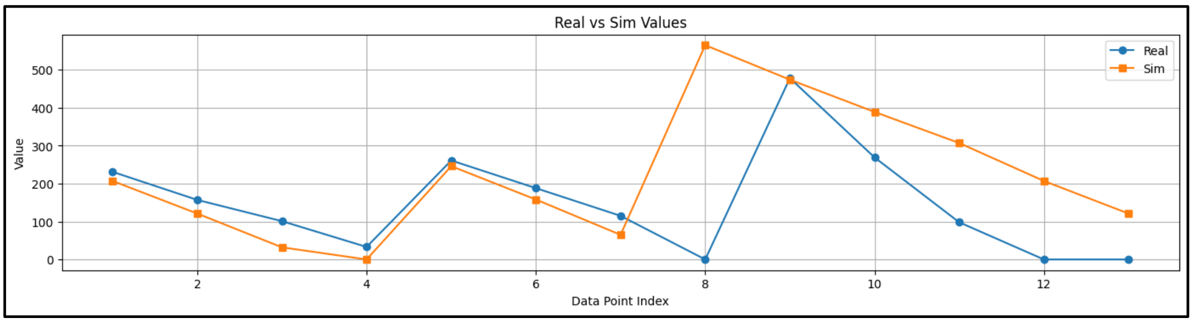
The Traditional Queueing System tends to underestimate or overestimate with larger deviations, as reflected in higher absolute differences and a t-test value of -1.5075, which suggests a moderate discrepancy from the real data. In contrast, the Time-Based and Event-Triggered Control approach produces simulation results closer to the real values, showing smaller absolute differences overall and a less extreme t-test value of -0.79442. The statistical test (df=12, α=0.05) yields a p-value of 2.179, falling within the acceptance range of the null hypothesis (H_0: no difference), indicating that the Time-Based and Event-Triggered approach aligns better with actual data and improves simulation accuracy compared to the traditional method as Table 6.

Table 6 The comparison of t-test values

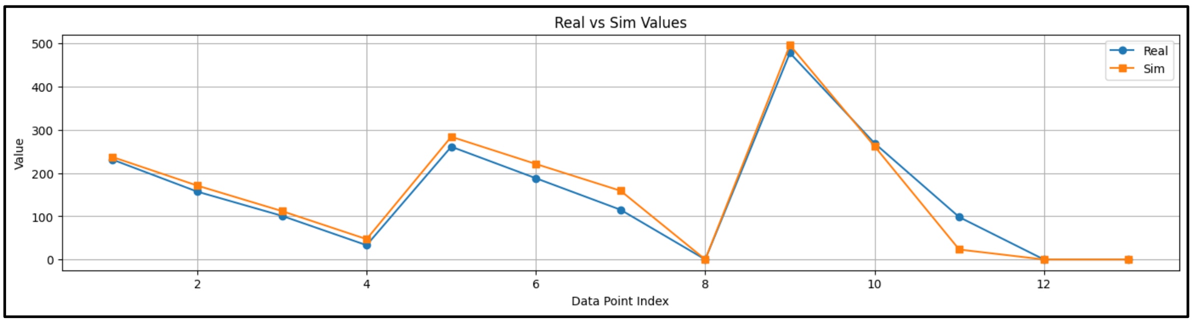
No.	Real	Sim Using Traditional Queueing System	Diff	Sim Using Time-based and Event-triggered control	Diff
1	231	207	24	237	-6
2	157	121	36	171	-14
3	101	32	69	112	-11
4	33	0	33	47	-14
5	261	246	15	284	-23
6	188	158	30	221	-33
7	115	65	50	159	-44
8	0	565	-565	0	0
9	478	474	4	496	-18
10	269	389	-120	262	7
11	98	307	-209	23	75
12	0	207	-207	0	0
13	0	121	-121	0	0
t-test		-1.5075		-0.79442	

When comparing the simulation results using the Traditional Queueing System and the Time-Based and Event-Triggered Control approach, the Time-Based and Event-Triggered method demonstrates superior performance in terms of accuracy. The Mean Absolute Error (MAE) for the Traditional Queueing System is significantly higher at 114.08, whereas the Time-Based and Event-Triggered Control approach achieves a notably lower MAE of 20.92, indicating a closer alignment with real values. Similarly, the Mean Absolute Percentage Error (MAPE) for the traditional method is 52.56%, reflecting large relative deviations from actual data, while the Time-Based and Event-Triggered Control approach reduces the MAPE to 10.13%. These results clearly show that incorporating Time-Based and Event-Triggered Control into the simulation substantially improves accuracy and reliability over the traditional queuing system model.

Moreover, Figure 10(a) and Figure 10(b) illustrate the "Real vs Sim Values" for different control strategies. Figure 10(a) depicts a Traditional Queueing System, where the "Sim" values (orange line with squares) show significant deviations from the "Real" values (blue line with circles) at several data points, particularly around data point 8 where the "Sim" value peaks significantly higher than the "Real" value, and at data points 10, 11, and 12 where the "Sim" values remain elevated while "Real" values drop to zero. In contrast, Figure 10(b) showcases a Time-Based and Event-Triggered Control Approach, demonstrating a much closer alignment between the "Real" and "Sim" values across all data points. The "Sim" values in Figure 10(b) accurately track the fluctuations and trends of the "Real" values, including the sharp peak at data point 9 and the subsequent decline. This direct comparison highlights the superior performance of the time-based and event-triggered control approach in replicating real-world system behavior compared to the traditional queueing system, suggesting enhanced accuracy and responsiveness in control.



(a)



(b)

Fig. 10 Comparison of control strategies: (a) Traditional queueing system; (b) Time-based and event-triggered control approach

5. Discussion

The dynamic behavior of each scenario reflects the interplay between the seasonal order quantity and the resulting inventory performance in terms of "Releasing," "Empty," and "Last Stock" values. Scenario C, the base case with 600 seasonal units, shows a moderate release rate (79.39%)

and a relatively high percentage of time without inventory (20.61%), indicating insufficient stock to meet demand. As the seasonal order quantity increases from Scenario 1 to 7 (650 to 950 units), the "Releasing" percentage steadily improves while the "Empty" percentage decreases, showing better alignment with demand and fewer stockouts, but still with no excess stock (Last Stock = 0). In Scenario 8, ordering 1000 units yields a major improvement (93.04% releasing, only 6.96% empty), yet introduces leftover inventory (Last Stock = 32), indicating a transition from just meeting demand to slight overstocking. This overstock trend continues as the normal order is increased in Scenarios 9–12 (350–500 units), improving "Releasing" to 99.38% and reducing "Empty" to near-zero, though Last Stock also rises, peaking at 37 units. Scenario 13, with 550 units normal and 1000 seasonal, achieves 100% service level (no empty period) but results in the highest leftover inventory (Last Stock = 72), suggesting excess ordering. Scenario 14, reducing seasonal quantity slightly to 950, still maintains nearly perfect service (99.87% releasing) and completely avoids excess inventory (Last Stock = 0), representing the most balanced and efficient scenario among all.

The comparison between the Traditional Queueing System simulation and the Time-Based and Event-Triggered Control approach reveals significant differences in accuracy when predicting real values. The Traditional Queueing System exhibits larger deviations (t-test: -1.5075, MAE: 114.08, MAPE: 52.56%), often underestimating or overestimating. Conversely, the Time-Based and Event-Triggered Control approach produces results notably closer to real values (t-test: -0.79442, MAE: 20.92, MAPE: 10.13%). The statistical test (df=12, α=0.05, p-value=2.179) further supports that the Time-Based and Event-Triggered method aligns better with actual data, significantly improving simulation accuracy and reliability over the traditional queuing system model.

The proposed demand-switching algorithm significantly contributes to the field by operationalizing both event-triggered control strategies and time-triggered logic within a discrete-event simulation environment. Unlike prior studies that primarily focused on theoretical or mathematical control frameworks, this research demonstrates a practical application of these concepts in a retail inventory context. Zhang et al. [12] introduced an event-triggered control algorithm for discrete-time delay systems where actions are only taken when a measurement error exceeds a threshold, thereby reducing unnecessary control updates. Similarly, Xu et al. [13] proposed a segmentation-based event-triggered mechanism that activates control actions only when the system state enters specific regions. While these studies optimize control frequency and computational efficiency, they are mainly centered on the stability and performance of technical control systems.

In contrast, this study adopts the event-triggered concept to switch demand types only when specific time-based conditions are met, thereby optimizing inventory flow without constant monitoring or manual intervention. Moreover, Hashimoto and Dimarogonas [14] highlighted the role of time-triggered logic in ensuring system behavior compliance with temporal logic specifications, particularly in resource-constrained networked systems. Maity and Baras [15] advanced this concept by integrating temporal logic with event-triggered frameworks to guarantee trajectory constraints are met while minimizing communication loads. Building upon these ideas, this study implements time-triggered logic through simulation time checkpoints that activate or deactivate FlexSim ports at the beginning and end of seasonal demand periods. By combining these two approaches—event-based activation conditions with specific time-based triggers—the proposed method ensures precise, autonomous switching between normal and seasonal demand modes in a looped cycle, including reversion to normal demand after seasonal peaks.

6. Conclusion

This study highlights the importance of optimizing Reorder Points (order quantity) to manage seasonal demand fluctuations in small fashion retail businesses, specifically in the case of a football jersey shop that experiences demand surges during competition months. By using FlexSim simulation, various inventory scenarios were tested to identify the most effective order quantity strategy that balances demand fulfillment and inventory efficiency. The results demonstrate that increasing the seasonal order quantity significantly improves the Releasing percentage while reducing stockouts (Empty occurrences). However, excessively high order quantity values lead to inventory accumulation (Last Stock), reinforcing the need for a balanced approach.

The findings confirm that Scenario 14 (550/950 order quantity) provided the most optimized inventory strategy, achieving 99.87% Releasing, only 0.13% Empty, and no excess inventory, making it the best approach to minimize both stockouts and overstocking. These results align with previous research emphasizing the importance of data-driven inventory adjustments and simulation-based decision-making in optimizing stock levels. Furthermore, this study fills a critical research gap by applying simulation-based inventory optimization in small-scale retail businesses, a field often overlooked in prior studies. The integration of automated control mechanisms in FlexSim, such as port-closing and opening triggers, further enhances model accuracy, making it a valuable tool for dynamic inventory management.

In terms of practical contributions, this research suggests that discrete-event simulation software (i.e., FlexSim) can be utilized by the SMEs in the retail industry to improve inventory optimization. As the SMEs usually lack operational resources due to financial restrictions, they may not have the capabilities to deal with the lost sales opportunities from stockouts or excessive inventory holding costs. Therefore, instead of using conventional forecasting methods, this study proposes that the SMEs and practitioners in the retail industry may adopt FlexSim simulation as a practical solution to identify the optimal order levels for products that are subject to seasonal demand fluctuations by experimenting with different possible order strategies in a risk-free environment and gain clear visual insights into the outcomes.

Despite the benefits, the use of FlexSim with trial-and-error for control system development has limitations. FlexSim may struggle to model complex real-world dynamics, such as intricate interactions or non-linear behaviors, creating a gap between simulation and reality. Trial-and-error is also time-consuming and resource-intensive for complex systems, often requiring numerous costly runs without guaranteeing a globally optimal solution, and can lead to suboptimal, less robust outcomes in practice. However, within FlexSim, the trial-and-error facilitates the execution of multiple replications with random variables, allowing for a deep understanding of the system's behavior under various conditions and identifying robust solutions without disrupting real-world operations, which are challenging for static analytical models.

Future research should focus on enhancing inventory optimization by integrating AI-based demand forecasting, allowing for more precise predictions of seasonal fluctuations and improving order quantity strategies. Secondly, as this study only focuses on one stock-keeping unit (SKU), expanding the research to multi-product inventory management would provide deeper insights into handling diverse demand patterns while incorporating variable supplier lead times in the simulation model, which can improve real-world applicability and supply chain resilience. Thirdly, conducting a cost-benefit analysis will help businesses balance financial efficiency with inventory performance. Lastly, testing simulation-based inventory control in other retail sectors, such as electronics, groceries, or pharmaceuticals, can validate the effectiveness of FlexSim across different industries, ultimately refining inventory management strategies for broader applications.

7. References

[1] S. Chopra and P. Meindl, *Supply Chain Management: Strategy, Planning, and Operation*, 6th ed. NJ, USA: Pearson, 2016.

[2] S. Nahmias and T. L. Olsen, *Production and Operations Analysis*, 7th ed. Long Grove, IL: Waveland Press, 2015.

[3] E. A. Silver, D. F. Pyke, and D. J. Thomas, *Inventory Management and Production Planning and Scheduling*, 3rd ed. New York, NY: Wiley, 1998.

[4] P. H. Zipkin, *Foundations of Inventory Management*. New York, NY: McGraw-Hill, 2000.

[5] Y. Zhou, X. Shen and Y. Yu, "Inventory control strategy: Based on demand forecast error," *Modern Supply Chain Research and Applications*, vol. 5, no. 4, pp. 74–101, Aug. 2023, doi: 10.1108/MSCRA-02-2023-0009.

[6] J. Riezebos and S. X. Zhu, " Inventory control with seasonality of lead times," *Omega: The International Journal of Management Science*, vol. 92, Apr. 2020, Art no. 102162, doi: 10.1016/j.omega.2019.102162.

[7] R. Abbou, J. J. Loiseau, and C. Moussaoui, " Robust inventory control of production systems subject to uncertainties on demand and lead times," *International Journal of Production Research*, " vol. 55, no. 8, pp. 2177–2196, Dec. 2015, doi: 10.1080/00207543.2016.1214295.

[8] N. Azizi and S. Zolfaghari, "A simulation-based optimization approach for inventory control of perishable products," *Simulation Modelling Practice and Theory*, vol. 12, no. 7–8, pp. 505–519, Apr. 2004, doi: 10.1016/j.cor.2023.106270.

[9] A. M. Law, *Simulation Modeling and Analysis*, 5th ed. New York, NY: McGraw-Hill, 2014.

[10] A. R. Ravindran, D. T. Phillips, and J. J. Solberg, *Operations Research: Principles and Practice*, 2nd ed. Hoboken, NJ: Wiley, 2008.

[11] J. Olhager and M. Rudberg, "Linking manufacturing strategy decisions on process choice with manufacturing planning and control systems," *International Journal of Production Research*, vol. 40, no. 10, pp. 2335–2351, Nov. 2002, doi: 10.1080/00207540210131842.

[12] L. Zhang, Y. Zhou, and X. Liu, "Event-triggered control for discrete-time delay systems based on dynamic threshold strategy," *International Journal of Control*, vol. 95, no. 3, pp. 635–648, Aug. 2022, doi: 10.1016/j.automatica.2022.110688.

[13] J. Xu, H. Wang, and Y. Chen, "State-segmentation-based event-triggered control for discrete-time 2-D switched systems," *IEEE Transactions on Circuits and Systems II, Express Briefs*, vol. 71, no. 12, Dec. 2024, doi: 10.1109/CYBER59472.2023.10256539.

[14] K. Hashimoto and D. V. Dimarogonas, "Resource-aware networked control under temporal logic specifications," *IEEE Transactions on Control Network Systems*, vol. 6, no. 3, pp. 1108–1118, Sep. 2019, doi: 10.1007/s10626-019-00297-7.

[15] A. Maity and J. S. Baras, "Event-triggered control under temporal logic constraints," *IEEE Transactions on Automatic Control*, vol. 63, no. 8, pp. 2473–2480, Aug. 2018, doi: 10.23919/ACC.2018.8430964.

[16] FlexSim Software Products, Inc. (2024). *FlexSim Simulation Software*. Accessed: Jun. 11, 2025. [Online]. Available: https://www.flexsim.com

[17] The AnyLogic Company. (2025). *AnyLogic: Simulation Modeling Software Tools*. Accessed: Jun. 11, 2025. [Online]. Available: https://www.anylogic.com

[18] Rockwell Automation. (2025). *Arena Simulation Software*. Accessed: Jun. 11, 2025. [Online]. Available: https://www.arenasimulation.com

[19] Simul8 Corporation. (2025). *Simul8 Simulation Software*. Accessed: Jun. 11, 2025. [Online]. Available: https://www.simul8.com