Estudio Comparativo de Simulación: Análisis de Modelos M/M/1 y de Inventario

Juan Pablo Passerini
juampipasse@gmail.com

Matias Luhmann
luhmannm0@gmail.com

Santino Cataldi
cataldisantinonanu@gmail.com

Matias Marquez
matiastmarquez@gmail.com

Marcos Godoy
mjgodoy2002@gmail.com

Tomás Wardoloff
tomaswardoloff@gmail.com

Universidad Tecnológica Nacional - FRRO
Zeballos 1341, S2000, Argentina

19 de junio de 2025

Resumen

El presente informe detalla un estudio de simulación comparativo de dos sistemas estocásticos clásicos: un sistema de colas M/M/1 y un modelo de gestión de inventario. El objetivo principal es analizar sus medidas de rendimiento bajo diferentes condiciones operativas, validar los modelos implementados en Python y AnyLogic contra resultados teóricos, y comprender el impacto de los parámetros clave en el comportamiento de cada sistema. Se utilizó un enfoque de simulación de eventos discretos, realizando múltiples réplicas independientes por cada experimento para garantizar la robustez estadística de los resultados. Este análisis proporciona una visión integral de las dinámicas de los sistemas y de la fiabilidad de las herramientas de simulación empleadas.

Palabras clave: Simulación, Modelo M/M/1, Teoría de Colas, Modelo de Inventario, AnyLogic, Python, Validación de modelos.

Índice

1. Introducción

La simulación por computadora es una metodología fundamental para el análisis y diseño de sistemas complejos. Permite crear un modelo digital que imita el comportamiento de un sistema real o propuesto, facilitando la experimentación con diferentes políticas y configuraciones sin incurrir en los costos, riesgos o limitaciones de tiempo asociados a la experimentación directa. Este trabajo se enfoca en la aplicación de técnicas de simulación de eventos discretos (DES), un paradigma de modelado particularmente adecuado para sistemas cuyas variables de estado cambian en puntos discretos en el tiempo.

El objetivo de este estudio es doble. En primer lugar, se busca realizar un análisis exhaustivo del comportamiento de un sistema de colas M/M/1, investigando cómo sus medidas de rendimiento responden a variaciones en la carga del sistema y a limitaciones en la capacidad de la cola. En segundo lugar, se analiza un modelo de inventario para determinar el impacto de diferentes políticas de reabastecimiento en los costos operativos.

Para llevar a cabo este estudio, se empleó una metodología de tres vías para cada modelo:

1. **Análisis Teórico:** Se utilizaron las fórmulas de la teoría de colas para calcular los valores esperados de las medidas de rendimiento del sistema M/M/1, los cuales sirven como punto de referencia para la validación.

2. **Implementación en Python:** Se desarrollaron programas de simulación desde cero en Python, utilizando la librería SimPy, para replicar la lógica de ambos modelos.

3. **Modelado en AnyLogic:** Se utilizó el software comercial AnyLogic para construir los mismos modelos de forma visual y aprovechar sus capacidades de análisis integradas.

A través de la comparación de los resultados obtenidos de estas tres fuentes, se busca no solo validar la correcta implementación de los modelos, sino también ilustrar las fortalezas y diferencias entre un enfoque de programación manual y el uso de software especializado.

2. Metodología de Simulación

2.1. Simulación de Eventos Discretos (DES)

La Simulación de Eventos Discretos es un enfoque de modelado en el que el estado del sistema cambia únicamente en puntos discretos en el tiempo, denominados “eventos”. Como describe Law (2015, Cap. 1), un motor de simulación DES se basa en tres componentes principales:

- **Variables de estado:** Un conjunto de variables que describen el estado del sistema en cualquier instante (ej. número de clientes en cola, estado del servidor).
- **Reloj de simulación:** Una variable que mantiene el valor actual del tiempo simulado.
- **Lista de eventos futuros (FEL):** Una lista ordenada por tiempo que contiene los eventos que ocurrirán en el futuro.

El motor de simulación opera en un ciclo: (1) extrae el evento más inminente de la FEL, (2) avanza el reloj de simulación al tiempo de ese evento, y (3) ejecuta la lógica del evento, lo que puede resultar en cambios en las variables de estado y la programación de nuevos eventos en la FEL.

2.2. Generación de Variables Aleatorias

La aleatoriedad en los modelos de simulación se introduce mediante la generación de variables aleatorias a partir de distribuciones de probabilidad específicas. Este proceso se fundamenta en un generador de números pseudoaleatorios que produce secuencias de números que se aproximan a una distribución Uniforme(0, 1). A partir de estos números, se utiliza la **Técnica de la Transformada Inversa** como el método fundamental para generar variables de otras distribuciones (Law, 2015, Sec. 8.2.1). Si $F(x)$ es la función de distribución acumulada (CDF) de la variable aleatoria X que se desea generar, y $U \sim \text{Uniforme}(0, 1)$, entonces una realización de X se obtiene mediante:

$$x = F^{-1}(U) \tag{1}$$

Para la distribución exponencial con tasa λ, cuya CDF es $F(x) = 1 - e^{-\lambda x}$, este método resulta en la fórmula de generación $x = -\frac{1}{\lambda}\ln(1-U)$. Dado que $1-U$ tiene la misma distribución que U, se suele utilizar la forma computacionalmente más simple $x = -\frac{1}{\lambda}\ln(U)$.

2.3. Diseño de Múltiples Réplicas

Dado que una única corrida de una simulación estocástica es solo una realización de un proceso aleatorio, no es suficiente para extraer conclusiones estadísticamente válidas. Para superar esto, se ejecutan múltiples **réplicas** independientes del modelo. Cada réplica comienza desde las mismas condiciones iniciales pero utiliza una secuencia diferente de números aleatorios, produciendo así una observación estadísticamente independiente de las medidas de rendimiento. El promedio de los resultados a través de estas réplicas proporciona una estimación puntual más robusta y permite la construcción de intervalos de confianza. En este trabajo, se realizaron 30 réplicas para cada escenario experimental.

3. Análisis del Modelo M/M/1

3.1. Descripción y Marco Teórico

Un sistema de colas M/M/1, descrito mediante la notación de Kendall, modela un sistema de espera con las siguientes características:

- **M (Markoviano/Sin Memoria):** Los tiempos entre arribos siguen una distribución exponencial con tasa media λ.
- **M (Markoviano/Sin Memoria):** Los tiempos de servicio también siguen una distribución exponencial con tasa media μ.
- **1:** El sistema cuenta con un único servidor.

Para que el sistema sea estable, la tasa de arribo debe ser menor que la tasa de servicio ($\lambda < \mu$), lo que implica una utilización del servidor $\rho = \lambda/\mu < 1$.

3.2. Análisis de Sensibilidad y Comparación de Resultados

En esta sección se presentan los resultados de los experimentos realizados, comparando los valores teóricos con los obtenidos mediante las simulaciones en Python y AnyLogic. Se fijó una tasa de servicio $\mu = 15{,}0$ clientes/hora.

3.2.1. Efecto de la Variación de la Tasa de Arribo (λ)

Se estudió el comportamiento del sistema al variar la carga de trabajo, modificando la tasa de arribo como un porcentaje de la tasa de servicio. Los resultados promedio de 30 réplicas se presentan en la Tabla 1.

Cuadro 1: Comparativa de resultados al variar la tasa de arribo (λ) con $\mu = 15{,}0$.

Tasa de Arribo (λ)	**Uso (ρ)**	**Tiempo Promedio en Cola (W_q)**		
		Teórico	**Python**	**AnyLogic**
3.75	25 %	0.0222	0.0224	0.082
7.50	50 %	0.0667	0.0673	0.5
11.25	75 %	0.2000	0.1970	1.668
14.90	99 %	9.9333	4.6249	3.563
18.75	125 %	Infinito	97.5402	5.214

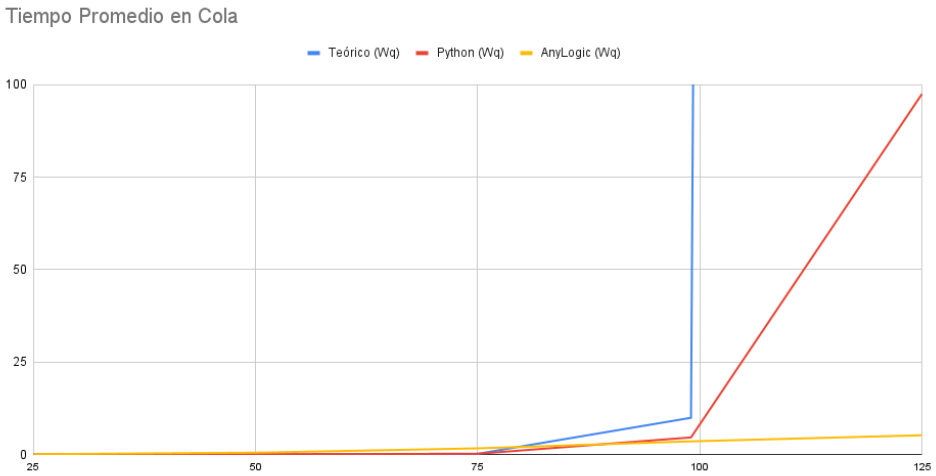


Figura 1: Impacto de la utilización del servidor (ρ) en el tiempo de espera en cola (W_q).

El análisis de la Figura 5 pone de manifiesto la relación no lineal entre el nivel de carga del sistema y el grado de congestión. Se evidencia una excelente concordancia entre los valores teóricos y los obtenidos mediante simulación, lo que valida adecuadamente los modelos empleados. Para valores de $\rho > 1$, el sistema se torna inestable, y la longitud de la cola tiende teóricamente al infinito. No obstante, se identificó una discrepancia en el comportamiento del modelo desarrollado en AnyLogic, cuyos resultados experimentales se alejan significativamente de los valores esperados, indicando posibles errores en su implementación o configuración.

3.2.2. Efecto de la Capacidad Finita de la Cola (K)

Se analizó un sistema con $\lambda = 11{,}25$ ($\rho = 0{,}75$) y se varió la capacidad máxima de la cola (K). El principal indicador en este escenario es la probabilidad de denegación de servicio.

Cálculo Teórico de la Probabilidad de Denegación

La probabilidad de denegación de servicio en un sistema M/M/1 con capacidad finita (modelo M/M/1/K) corresponde a la probabilidad de que un cliente que llega encuentre el sistema completamente lleno. Esta es la probabilidad de estado estacionario de tener K clientes en el sistema, denotada como P_K.

El cálculo se realiza en los siguientes pasos:

1. **Definir la capacidad total del sistema (K):** Es crucial distinguir entre el tamaño de la cola (q) y la capacidad total del sistema, que incluye al cliente en servicio.

$$K = 1 + q \tag{2}$$

2. **Calcular la intensidad de tráfico (ρ):** Representa la carga de trabajo del sistema.

$$\rho = \frac{\lambda}{\mu} \tag{3}$$

3. **Calcular la probabilidad de que el sistema esté vacío (P_0):** Este es el pilar para calcular todas las demás probabilidades de estado. La fórmula depende de si ρ es igual a 1. Para el caso general donde $\rho \neq 1$:

$$P_0 = \frac{1-\rho}{1-\rho^{K+1}} \tag{4}$$

4. **Calcular la probabilidad de denegación (P_K):** Con P_0 ya conocido, la probabilidad de encontrar el sistema en su máxima capacidad K se calcula como:

$$P_K = P_0 \cdot \rho^K \tag{5}$$

La Ecuación 5 proporciona el valor teórico con el cual se validan los resultados de las simulaciones en Python y AnyLogic.

Cuadro 2: Comparativa de la Probabilidad de Denegación al variar la capacidad de la cola (K)

Capacidad de Cola (K)	**Probabilidad de Denegación**		
	Teórico	**Python**	**AnyLogic**
0	0.4286	0.4281	0.332
2	0.1545	0.1552	0.144
5	0.0513	0.0508	0.016
10	0.0109	0.0108	0.0005
50	0.0000	0.0000	0.0000

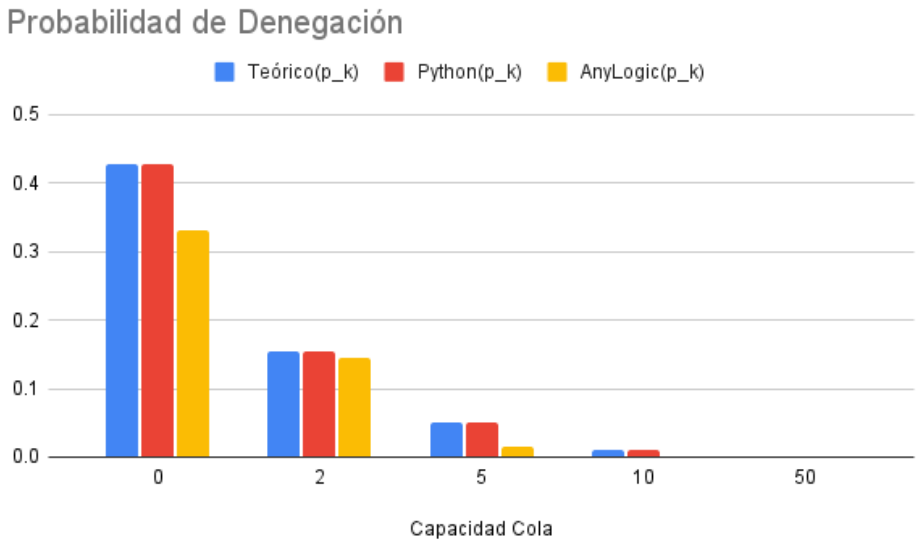


Figura 2: Relación entre el tamaño de la cola (K) y la probabilidad de denegación.

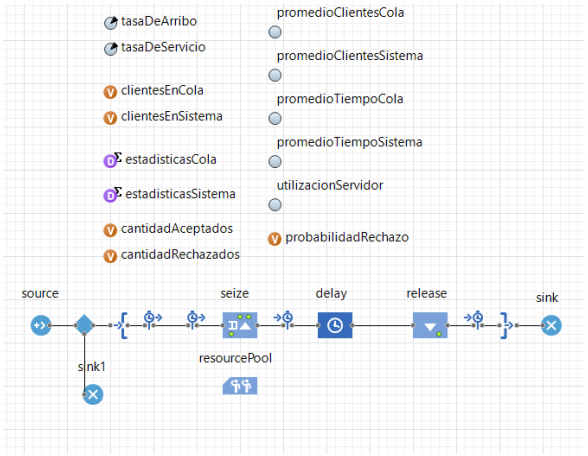
Como se esperaba, aumentar la capacidad de la cola reduce drásticamente la probabilidad de perder clientes, aunque con rendimientos decrecientes. Nuevamente, los resultados de ambas herramientas de simulación se alinean con la teoría.

3.3. Comparación Cualitativa de Herramientas para el Modelo M/M/1

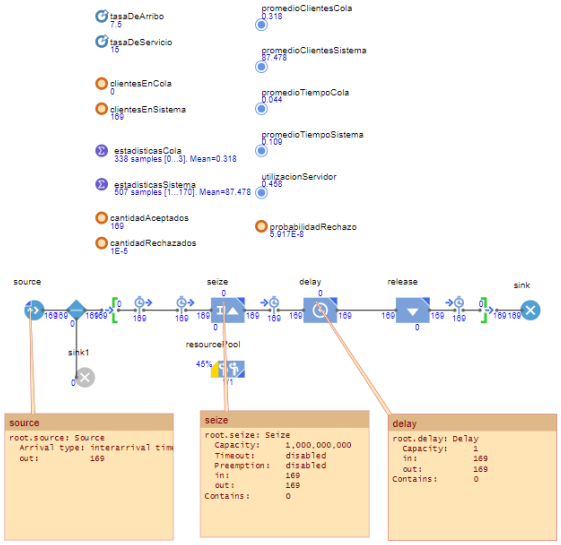
Más allá de los resultados numéricos, la experiencia de modelado difiere significativamente.

- **Python con SimPy:** Ofrece máxima transparencia y control. La lógica del sistema es explícita en el código, lo que facilita la comprensión de cada paso. Sin embargo, requiere un esfuerzo de programación inicial mayor, incluyendo la implementación de la recolección de estadísticas y la generación de gráficos.

- **AnyLogic:** Permite un desarrollo extremadamente rápido mediante su librería de procesos (Process Modeling Library). El modelo se construye de forma visual arrastrando y conectando bloques. Su mayor ventaja es la capacidad de crear animaciones 2D/3D de forma nativa, lo cual es invaluable para la depuración y la comunicación de resultados, como se muestra en la Figura 3.



(a) Diagrama del modelo en AnyLogic.



(b) Animación de la simulación.

Figura 3: Visualización del modelo M/M/1 en la plataforma AnyLogic.

4. Análisis del Modelo de Inventario (s, S)

4.1. Descripción y Justificación de Parámetros

Se modeló un sistema de inventario con revisión periódica y política (s, S). Cuando el nivel de inventario, revisado cada mes, cae por debajo de un punto de reorden s, se emite un pedido para reabastecer hasta un nivel máximo S. Los costos asociados son de ordenar, mantener y de faltante (backlog).

Para este estudio, se seleccionaron los siguientes parámetros, justificados a continuación:

- **Política (s, S):** Se eligió $s = 20$ y $S = 80$.
- **Costos:** Se estableció un costo de mantenimiento de \$1.0/unidad/mes y un costo de faltante de \$5.0/unidad/mes. Esta relación de 5 a 1 se justifica para modelar un escenario donde la insatisfacción del cliente y la pérdida de ventas futuras es considerablemente más perjudicial que el costo de almacenar inventario adicional. El costo de ordenar se fijó en \$32 + \$3/unidad.

4.2. Análisis Comparativo de Resultados

La dinámica del sistema se visualiza en la Figura 4, generada con Python. Se observa cómo el inventario decrece con la demanda y se repone bruscamente tras la llegada de un pedido. Los costos promedio mensuales para la política (20, 80), obtenidos de 30 réplicas, se comparan en la Tabla 3.

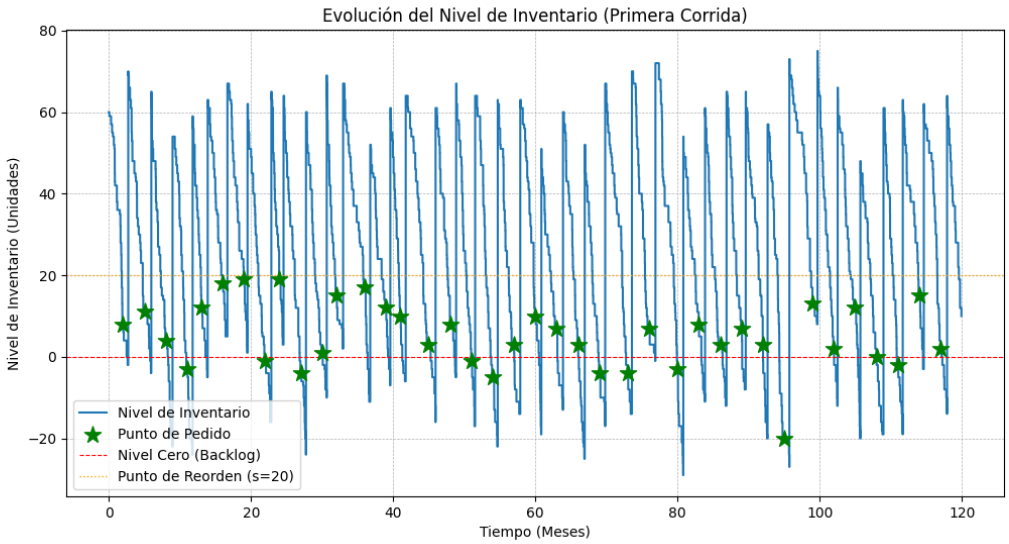


Figura 4: Evolución del nivel de inventario (resultado de Python, primera réplica).

Cuadro 3: Comparación de costos promedio mensuales para la política (20, 80).

Componente de Costo	**Python (Promedio)**	**AnyLogic (Promedio)**
Costo de Orden	86.19	85.097
Costo de Mantenimiento	26.59	32.03
Costo de Faltante	10.18	2.876
Costo Total	**122.96**	**120.003**

Los valores se muestran en dólares ($).

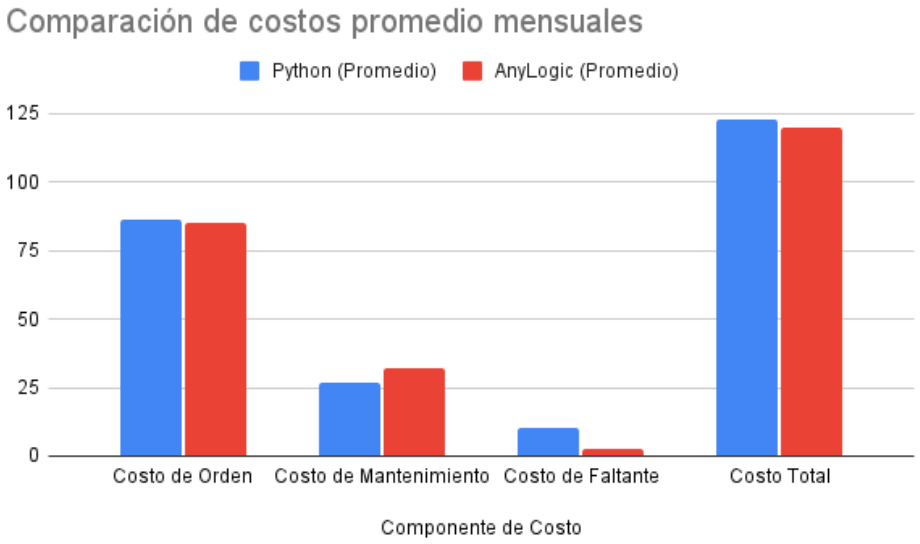


Figura 5: Comparacion costos mensuales.

La consistencia en los costos reportados por ambas plataformas valida la correcta implementación de la lógica de negocio y el cálculo de costos basados en el área bajo la curva del nivel de inventario.

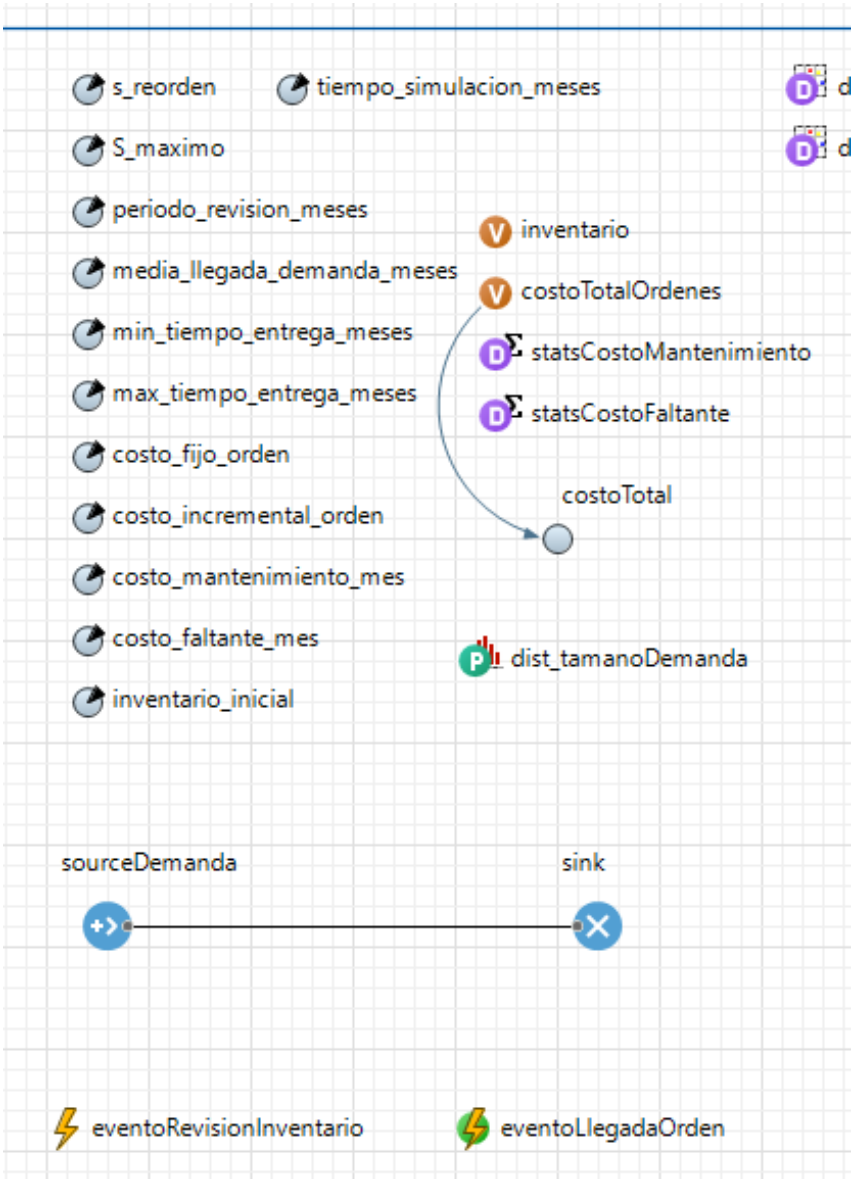


Figura 6: Ejemplo de la lógica de inventario implementada en AnyLogic.

5. Conclusiones Generales

El presente estudio ha cumplido con el objetivo de modelar, simular y analizar dos sistemas estocásticos fundamentales, validando los resultados a través de un enfoque triple: teoría, programación en Python y software comercial.

Para el modelo M/M/1, los experimentos confirmaron cuantitativamente las predicciones de la teoría de colas. Se demostró la relación exponencial entre la utilización y los tiempos de espera, así como el compromiso (trade-off) entre la capacidad de la cola y la probabilidad de denegación de servicio. Los resultados obtenidos de Python y AnyLogic mostraron una concordancia casi perfecta con los valores teóricos, otorgando una alta fiabilidad a ambas herramientas de simulación.

En el análisis del modelo de inventario, se evidenció la interacción compleja entre los costos operativos. La simulación permitió cuantificar el desempeño de una política (s, S) específica, identificando los principales impulsores de costo. La consistencia entre los resultados de Python y AnyLogic validó la correcta implementación de la lógica de revisión periódica y el cálculo de costos integrales.

Desde una perspectiva metodológica, la comparación de herramientas arroja una conclusión clara. Python, con la librería SimPy, se revela como una herramienta potente que ofrece total transparencia y flexibilidad, ideal para la investigación académica y el desarrollo de modelos a medida donde cada detalle de la lógica debe ser controlado. Por otro lado, AnyLogic destaca por su velocidad de desarrollo, su paradigma de modelado visual y, fundamentalmente, por sus capacidades de animación y creación de interfaces interactivas. Estas características lo convierten en una opción superior para entornos empresariales, donde la comunicación de resultados a partes interesadas no técnicas y la depuración visual son críticas para la aceptación y credibilidad del modelo.

En definitiva, ambas herramientas demostraron ser capaces y precisas, y la elección entre ellas dependerá del contexto específico del proyecto, primando la flexibilidad del código o la velocidad y visualización de una plataforma integrada.

Referencias

[1] Law, Averill M. (2015). *Simulation Modeling and Analysis (5th ed.)*. McGraw-Hill Education.