CAPÍTULO 5. DESARROLLO

Durante el presente capitulo se mostrará el proceso del desarrollo del sistema, el cual fue descrito de manera breve en el capitulo anterior; la manera en que el sistema funciona, así como una aplicación del mismo a una empresa comercial.

5.1 Desarrollo del Sistema Simulador

En el capítulo anterior se definió el proceso de desarrollo y debido a la simplicidad de las primeras dos etapas, es decir a la de la formulación de los objetivos y la definición de los eventos involucrados, se considera que ya han sido descritas satisfactoriamente.

La siguiente parte del proceso fue la creación del código del simulador con Demanda y tiempo de espera de envío determinísticos en el lenguaje Visual Basic. Lo anterior se logra partiendo de un algoritmo previamente realizado para describir la arquitectura del sistema, la cual quiere decir el cómo funciona el sistema de manera interna para lograr el objetivo de simular de manera correcta un sistema de inventarios, la arquitectura detallada será presentada más adelante en éste capítulo. Ya concluida esta parte del desarrollo, se prosigue a la etapa de la validación del correcto funcionamiento de éste, la cual es realizada a la par de la ejecución del programa. Para la validación se llevó a cabo la simulación de nueve casos manualmente: tres casos en los que no llegaba a existir carencia, tres donde existía carencia y se manejaba retraso de ventas y por último tres donde se manejaba pérdida de ventas. El resultado del Costo Total Anual de estas nueve

simulaciones fue exactamente el mismo al arrojado por el sistema simulador. Para más detalle de éstos resultados favor de dirigirse al Apéndice B de esta tesis.

Para complementar el simulador de manera que funcione con D y l probabilísticos se recurrió al generador de variables aleatorias de @Risk, esto debido a su rápido y buen funcionamiento ya que utiliza el método de la transformada inversa y generadores recursivos múltiples creados por Pierre L`Ecuyer[1] en los años noventas. Otra razón para la utilización de @Risk como complemento del sistema simulador es que debido a que la demanda y los tiempos de espera siempre son positivos es necesario que las distribuciones de probabilidad de estas variables sean truncadas de manera que el limite inferior sea igual a cero en la demanda y en los tiempos de espera sea igual a uno y para una mayor simplicidad y ahorro de tiempo en la ejecución del sistema @Risk es un complemento ideal para lograrlo. La validación del simulador con D y l probabilísticos consistió en realizar una simulación con demanda diaria igual a x y tiempo de espera igual a y, ambos constantes; lo anterior a la par de una simulación con demanda diaria con distribución uniforme (x,x) y tiempo de espera con distribución (y,y) los resultados de ambas simulaciones debían ser idénticos para indicar que el funcionamiento del simulador con parámetros probabilísticos era correcto.

[1] Las publicaciones de Pierre L´Ecuyer sobre este tema se encuentran en http://www.iro.umontreal.ca/~lecuyer/papers.html

5.2 Desarrollo del sistema optimizador

Al igual que en el proceso de desarrollo del simulador se partió primero de un algoritmo para el funcionamiento del optimizador para así proceder a la creación del código del mismo. Para comprobar su correcto funcionamiento, es decir, comprobar que realmente nos arroje los parámetros que hagan que el Costo Total Anual se minimice; seguimos el siguiente procedimiento: a la par de la ejecución se irán anotando el valor de los "vecinos" en cada movimiento y comprobar que efectivamente esta tomando al mejor de ellos en cada uno de estos movimientos. Este procedimiento es suficiente puesto que el correcto funcionamiento del simulador ya ha sido comprobado previamente y éste es el que genera los vecinos. Para más detalles del procedimiento favor de dirigirse al Apéndice C de ésta tesis.

5.3 Descripción del programa

Para realizar un análisis más completo dividiremos el estudio del programa en dos, primero se presenta la descripción de la sección del programa dedicado a la simulación. Como ya fue explicado anteriormente, ésta simulación es utilizada para obtener el costo total anual por llevar inventario. En segundo lugar se presenta la descripción del módulo restante que forma parte del programa, para observar el código completo véase el Apéndice E.

Es importante mencionar, que para la construcción del programa se asumió que, las empresas cierran sus puertas cada domingo, es decir, no son susceptibles de tener demanda ni de recibir pedidos en domingo; solamente de lunes a sábado. Por lo tanto, la

única operación que es necesaria de registrar cada domingo es el costo de almacenamiento que se genera en ese día. Por esta razón, el programa trabaja con 313 días que son los 365 días del año menos los 52 domingos respectivos de ese año.

El procedimiento a seguir, en general, es el mismo en los dos modelos de inventario incluidos en el programa excepto en un módulo, el módulo encargado de detectar si es necesario realizar un pedido. En el caso del modelo (s,Q) este módulo es llamado *Revision* y para el caso del modelo (s, S) este es llamado *RevisionSs*.

Para llevar a cabo la simulación se utilizaron nueve módulos. A continuación se explica la función que cubre cada uno de ellos dentro del programa.

Simula. La simulación se inicia en este módulo. Las funciones de esta subrutina son: establecer las condiciones iniciales para la simulación.

SigEvento. Se encarga de guardar en los vectores ATED y RECEPAUX, la fecha en la cual se producirá la siguiente demanda y recepción del pedido, respectivamente. Además de avanzar el reloj hacia la fecha del evento más próximo.

ChecaReloj. Determina cual es el evento que sigue a partir de la fecha que determina *SigEvento*. Si hay varios eventos en esa fecha, determina en base a prioridades el orden que deben seguir dichos eventos. Si el reloj marca un múltiplo de seis querrá decir que es

sábado y este módulo es el encargado de agregar el costo de almacenamiento en que incurrirá el inventario el día domingo, es decir, este costo se agrega por adelantado.

GeneraD. Es el módulo encargado de simular la cantidad demandada, cuando ésta sea estocástica. Las cantidades son generados de acuerdo a la distribución seleccionada y los parámetros introducidos por el usuario.

GeneraTEP. Tiene la función de simular el tiempo de espera de los pedidos realizados.

Demanda. Este módulo actualiza el inventario final (INVF) de acuerdo a la cantidad demandada y calcula el costo por el almacenamiento del inventario por ese día. La variable AGOT contabiliza el número total de demanda que no pudo ser satisfecha por agotamiento de productos, en el caso de pérdida de ventas o la cantidad de demanda que tuvo que ser satisfecha en una fecha posterior debido al agotamiento de productos en el caso de ventas pendientes, es decir, esta variable guarda el número acumulado de agotamientos. La variable AGO almacena el número de agotamientos que existen entre la recepción de un pedido y otro, el uso de esta variable está en el módulo de *Recepción*.

Recepción. Actualiza el inventario físico después de la recepción de un pedido. Si se simula un caso de pedidos atrasados, se satisface primero la demanda de pedidos atrasados. Si el número de unidades de pedidos atrasados es mayor a la cantidad recibida no aumenta el inventario físico, en caso contrario, aumenta de acuerdo a la diferencia

entre estas dos variables, es decir, cantidad recibida Q y unidades de pedidos atrasados denotado por la variable AGO.

Revision. Realiza la operación de revisar el nivel del inventario físico. Si el nivel de inventario es menor al punto de repedido y además no existe otro pedido en cola se efectúa el pedido, se calcula el momento de su recepción. En caso contrario, no se efectúa ningún pedido.

RevisionSs. Realiza una función adicional a *Revisión*, una vez que se ha determinado que se debe realizar un pedido, calcula el valor de la cantidad a ordenar; esta cantidad es calculada quitando de la cantidad S el inventario físico en ese momento. La cantidad S es introducida por el usuario o es determinada por el programa dependiendo si lo que desea el usuario es únicamente hacer una simulación o si el usuario desea obtener el valor que minimize el costo anual de inventario.

CostoTotal. Determina el costo total anual de hacer pedidos y lo guarda en la variable COT, también determina el costo total anual de carencia y lo guarda en la variable CCT. Finalmente, suma los tres costos en los que puede incurrir la empresa por llevar un inventario, obteniendo así, el costo total anual por llevar inventario denotado por la variable CTA. Este costo será mandado a llamar por el módulo *Optimiza*.

En la Figura 5.1 se muestra el diagrama de flujo del funcionamiento completo del programa de simulación.

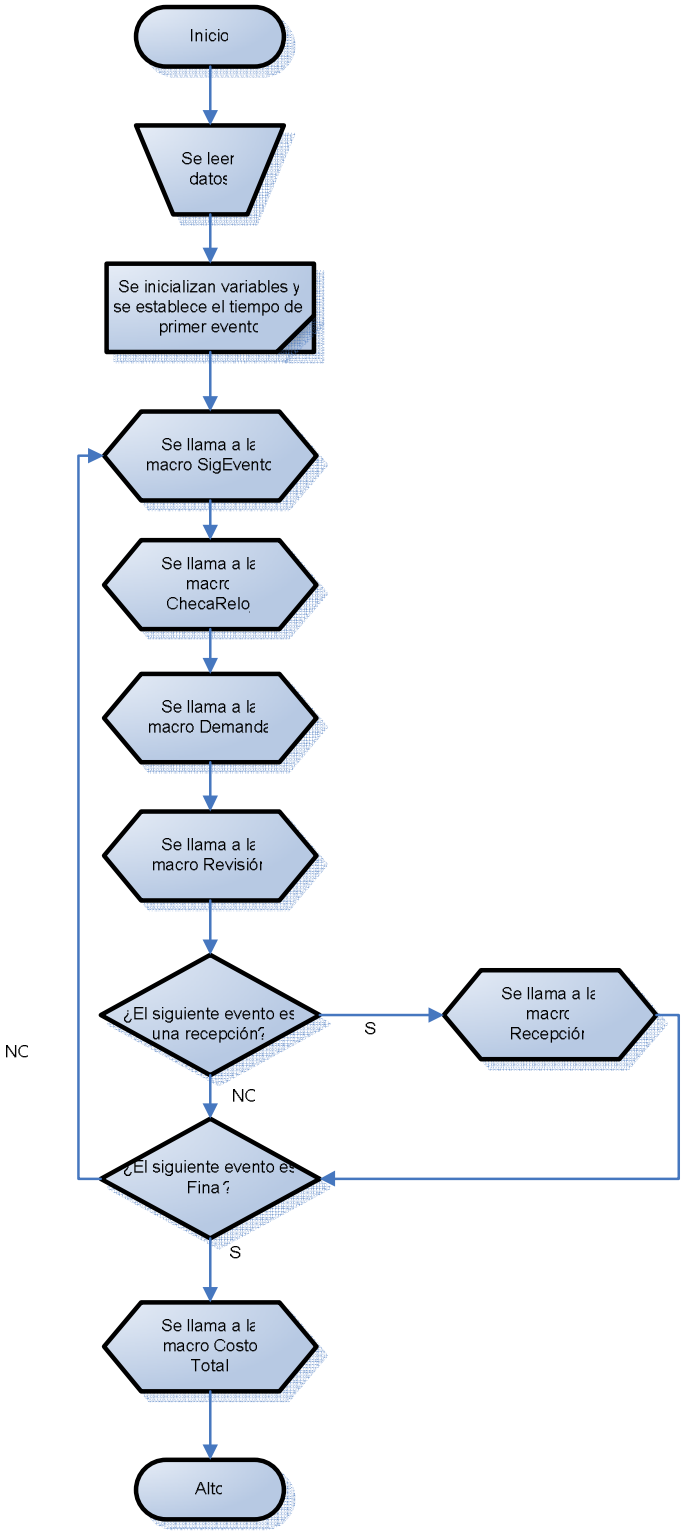

Figura 5.1 Arquitectura del simulador
Fuente: Elaboración propia

El programa principal es denominado ***Optimiza*** y es el encargo de llevar a cabo el proceso de optimización, es decir, en este módulo puede observarse cada uno de los pasos de Búsqueda Tabú descritos en el capítulo 2.

El proceso de optimización en este programa se inicia, calculando la Q y s iniciales, o la S y la s iniciales dependiendo del modelo seleccionado por el usuario. La siguiente operación que realiza es inicializar las variables necesarias para la iteración cero. A partir de la primera iteración, el programa genera el valor del costo total anual por llevar inventario de los correspondientes vecinos de la solución actual. El programa está configurado para generar ocho vecinos en cada iteración como se muestra en la Figura 5.2.

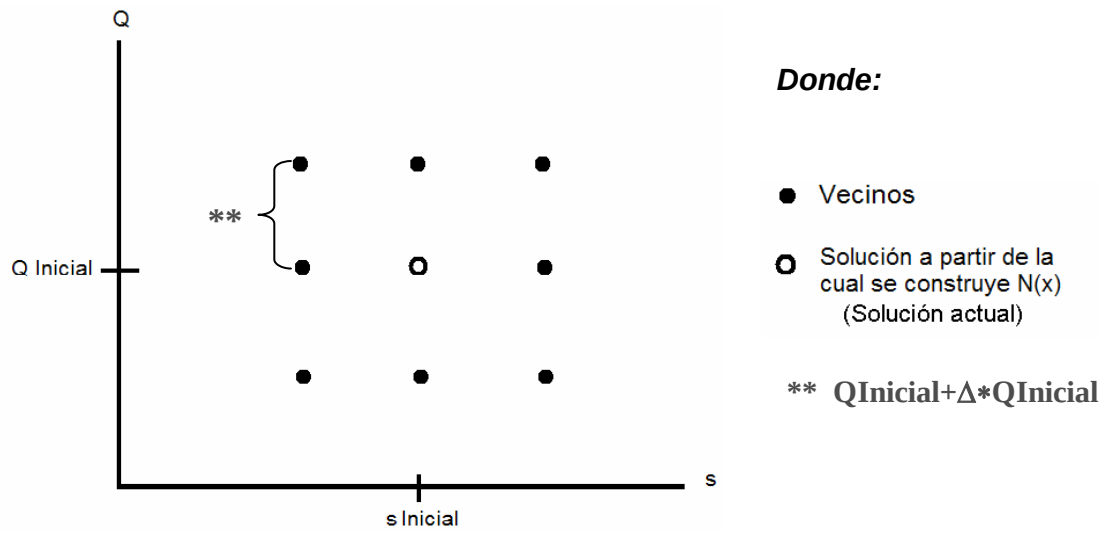

Figura 5.2 Proceso de generación de los ocho vecinos en cada iteración
Fuente: Elaboración propia

En caso de que el tipo de problema que aborde el programa sea estocástico, es decir, que sea seleccionado D o l estocástico, para cada vecino el programa mandará a llamar 35 veces la subrutina simula con el fin de poder obtener un promedio de esas 35

simulaciones, si desea que el programa funcione con mayor rapidez le recomendamos leer el *Manual de usuario* donde se explica como cambiar el número de iteraciones en cada vecino.

Una vez que se han calculado los ocho vecinos, se manda a llamar a la subrutina BubbleSort, la cual ordena esos valores de menor a mayor. La subrutina ChecaTabu verifica que la Q y la s correspondientes al valor más pequeño obtenido por BubbleSort no sea Tabú o la S y la s dependiendo del modelo seleccionado. Si es Tabú verifica el criterio de aspiración. Finalmente la lista tabú es actualizada. Este proceso se realiza dependiendo del número de iteraciones que al iniciar el usuario haya elegido al realizar la última de éstas el sistema arrojará en la pantalla de resultados el mejor costo obtenido junto a los valores que lo generan. El funcionamiento completo lo podemos observar en la Figura 5.3. Es muy importante que antes del primer uso del sistema, el usuario realice una lectura minuciosa del *Manual de usuario*(Apéndice F); esto con el fin de facilitar su correcto uso.

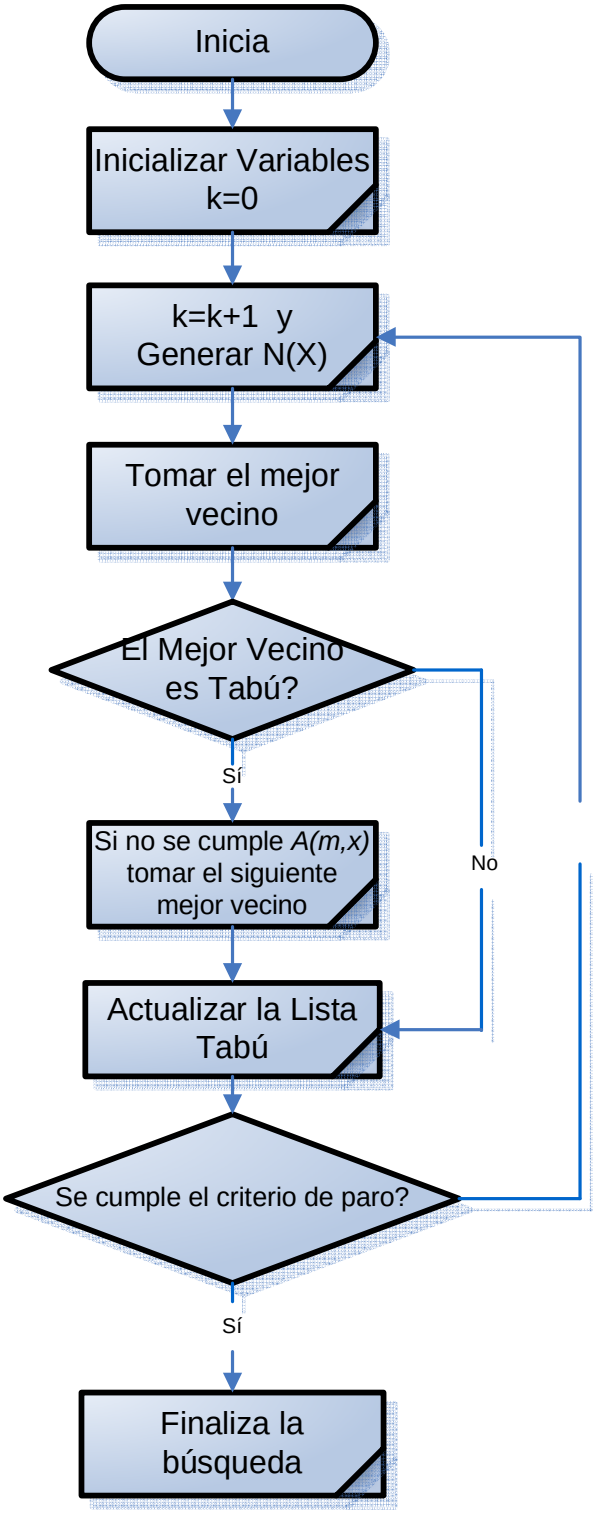

Figura 5.3 Arquitectura del sistema de optimización
Fuente: Elaboración propia

5.4 Aplicación del sistema a la empresa Orto-Istmo

Debido a que se deseaba aplicar el sistema a una empresa real, la empresa *Orto-Istmo* proporcionó la información necesaria para este fin. Esta es una empresa comercializadora de dispositivos ortopédicos con cede en la ciudad y puerto de Salina Cruz, en el estado de Oaxaca; el producto principal que comercializa son los zapatos ortopédicos infantiles, cuyo costo de mantenimiento de inventario será el objeto a analizar.

El análisis de los datos proporcionados por la empresa dio como resultado la información que se encuentra en la tabla 5.1. Para más detalle del proceso de análisis de los datos favor de dirigirse al Apéndice D.

Tabla 5.1 Datos de Orto Istmo

Costos	
Co	40
Cc	103.028
Ca	30
Variables	
Q	300
s	100
Pérdida de Ventas	
Distribuciones	
D	BinNeg(4,0.67641)
l	Triangular(15,66,66)

Fuente: Elaboración Propia

Con su política actual la empresa gastaba 5849.956 en mantener su inventario anualmente. El resultado obtenido tras introducir los datos en el sistema es Q= 118, s=112 con un Costo Total Anual de $3982.114 por lo que se recomienda a la empresa implemente estos resultados en su gestión de inventarios ya que adoptar esta política de inventario representaría un ahorro de 1867.84195 un 31.929 %.